Learning Representations for Graph Signals

Pascal Frossard, EPFL

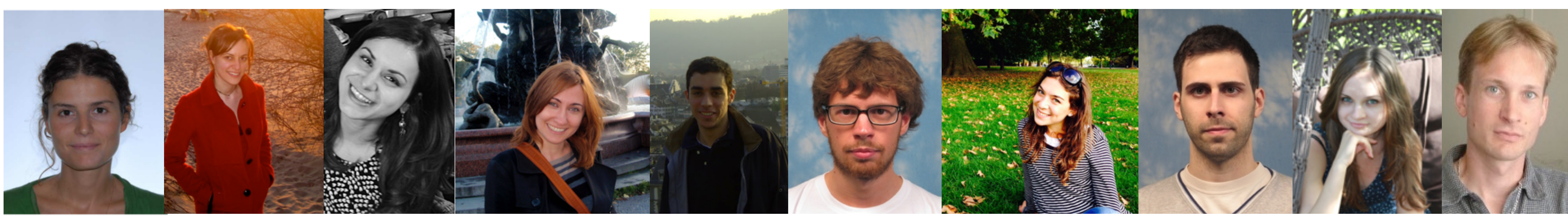
XRCE, Grenoble

June 9th, 2016

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

LTS4 in short

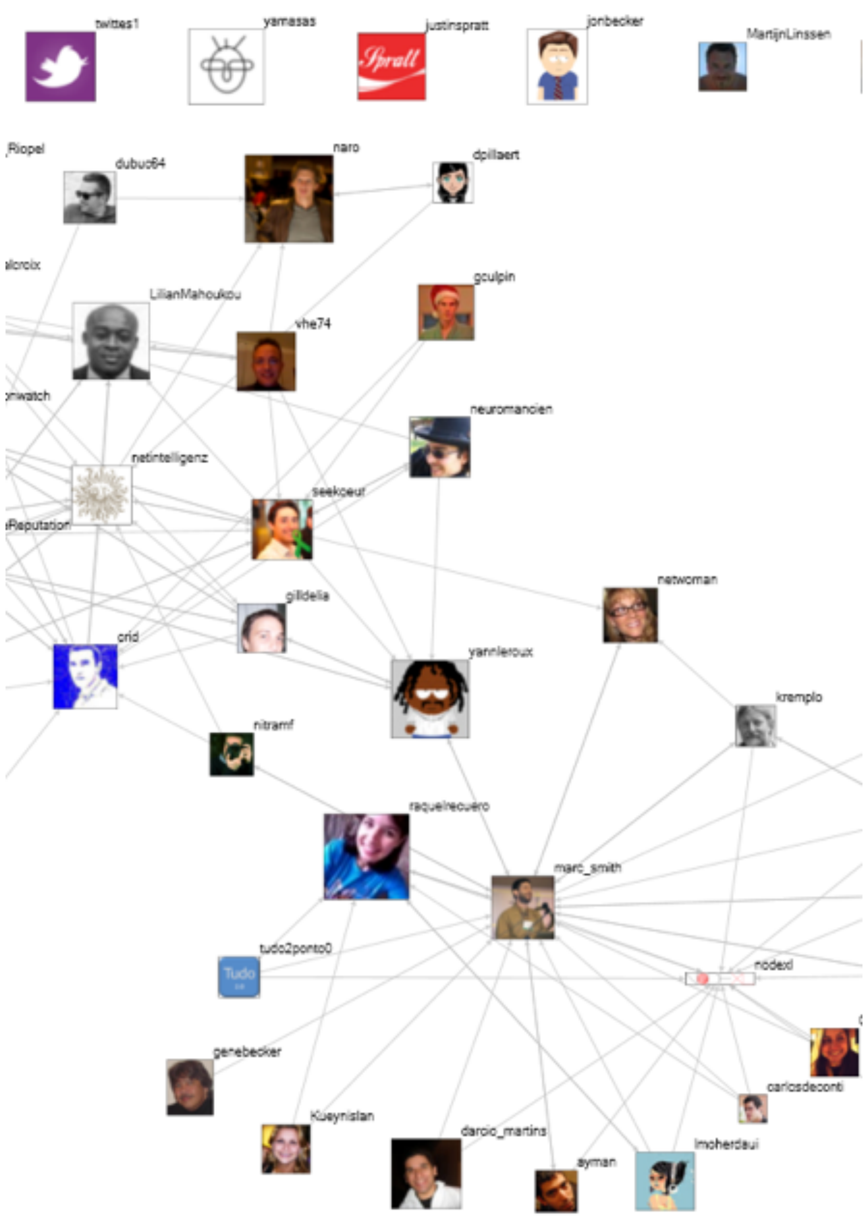
- Research group
  - 6 postdocs, 8 PhD students, several MSc students and interns
- Main funding sources
  - Nokia, Cisco, IBM, Google
  - Swiss CTI/KTI, EU FP7
  - Swiss NSF



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

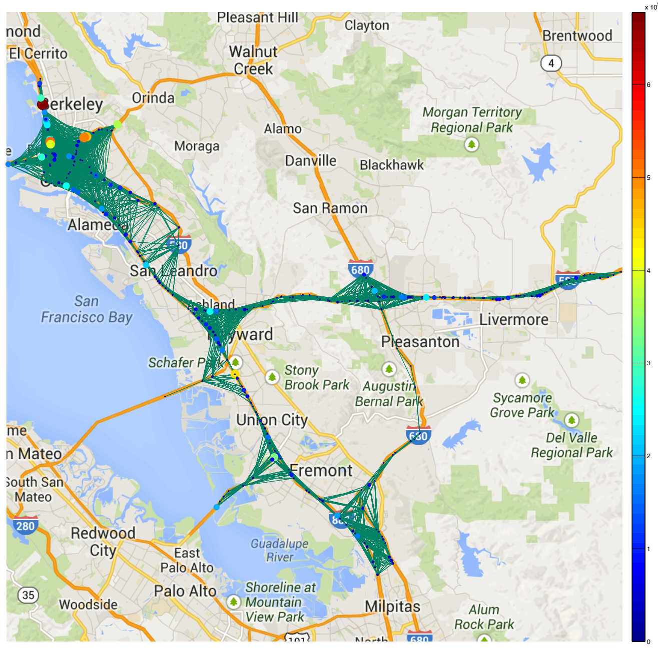
Main research topics

- Image processing
  - Computational imaging, 3D
  - Image analysis and classification
  - Immersive communication
- Distributed signal processing
  - Vision sensor networks, adaptive communication systems
- Graph Signal Processing
  - Analysis of network data (computer, social, traffic, brain networks…)



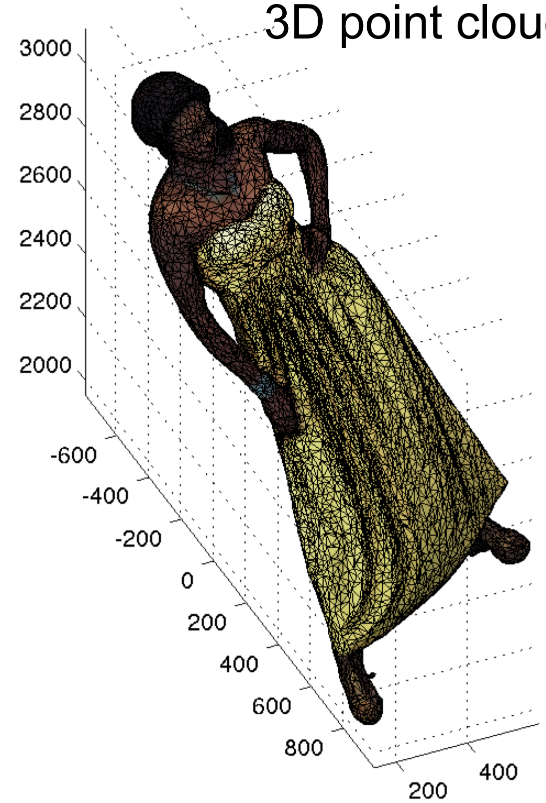
Social network data

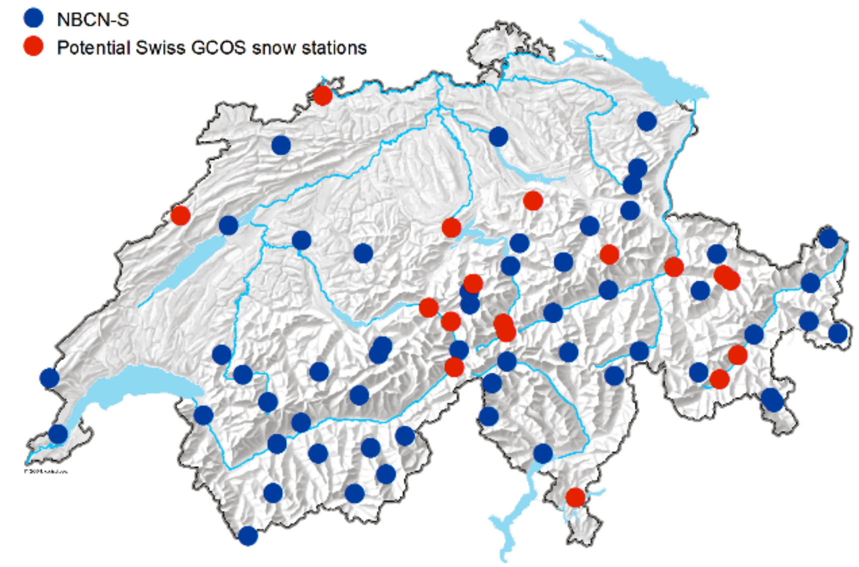
Structured data



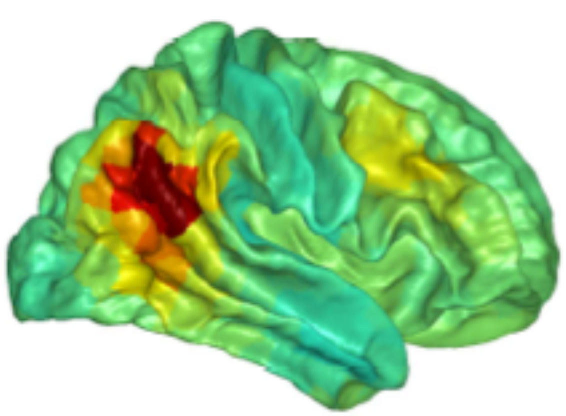
Traffic bottlenecks

3D point clouds



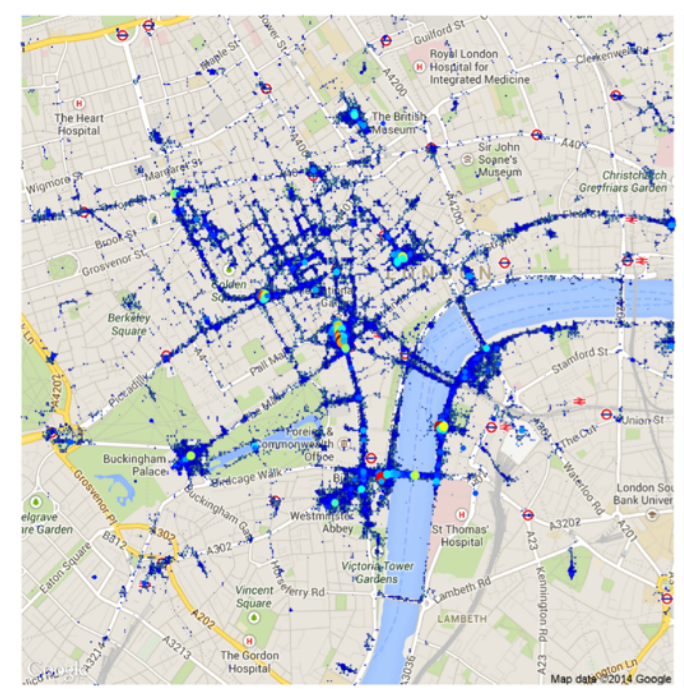


Sensor networks



Brain signals

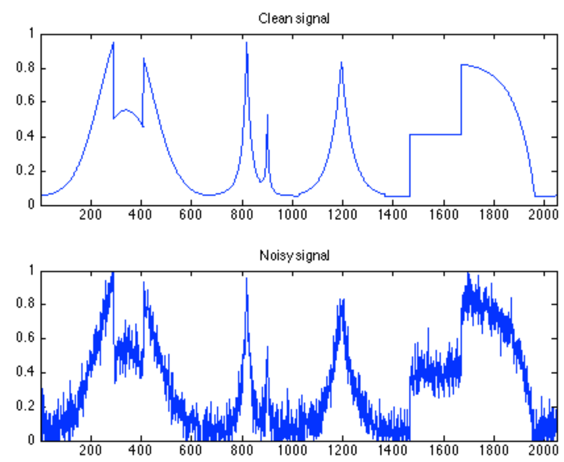
Social networks

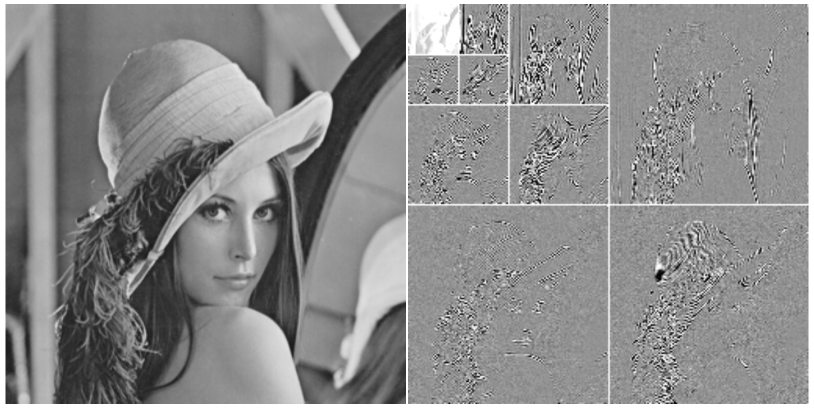


Mobility patterns

Structured, but irregular data …

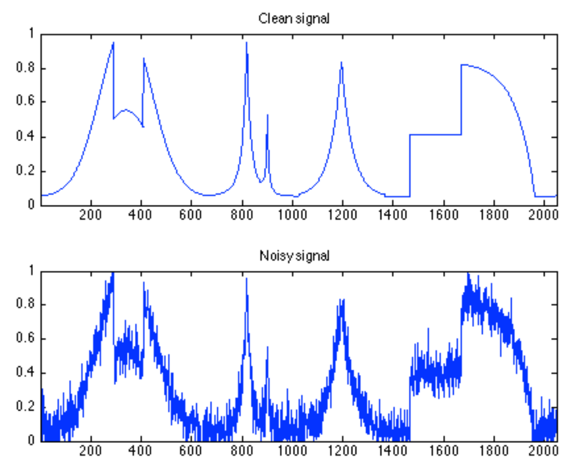
- Traditional signal processing in Euclidean space

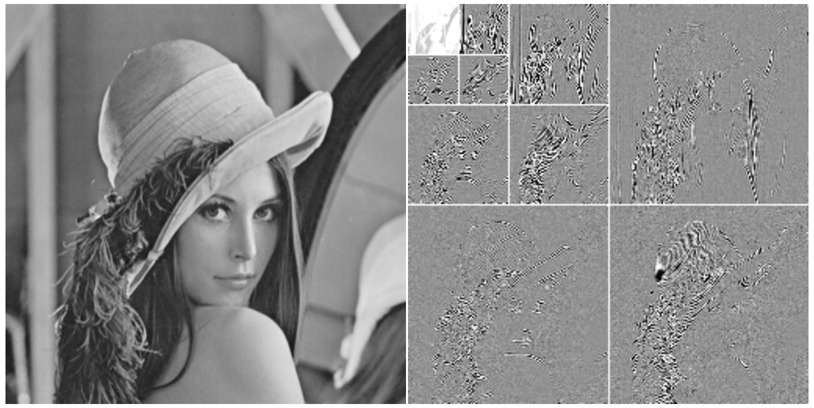




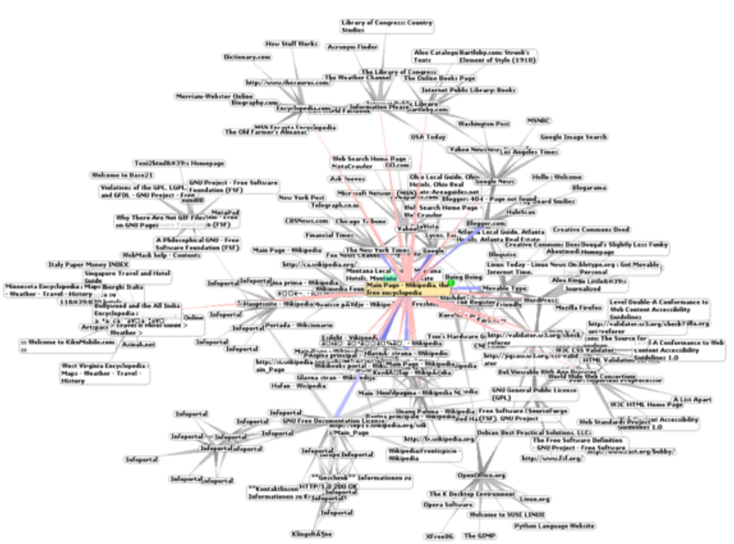
Structured, but irregular data …

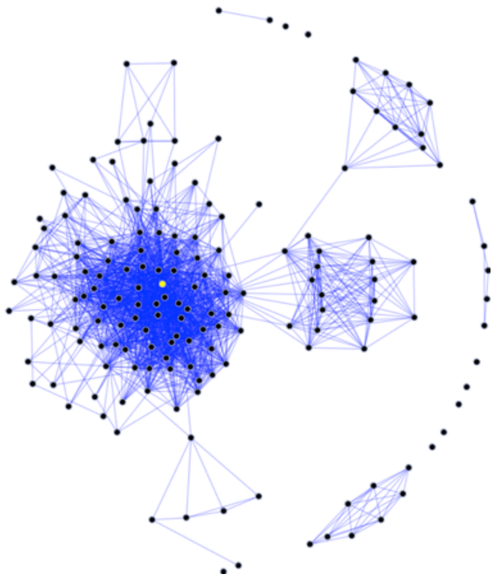
- Traditional signal processing in Euclidean space





- Irregular (graph) structures: new challenges for signal processing?





Acknowledgements

Dorina Thanou

Xiaowen Dong

David Shuman

Pierre Vandergheynst

Phil Chou

Antonio Ortega

Sunil Narang

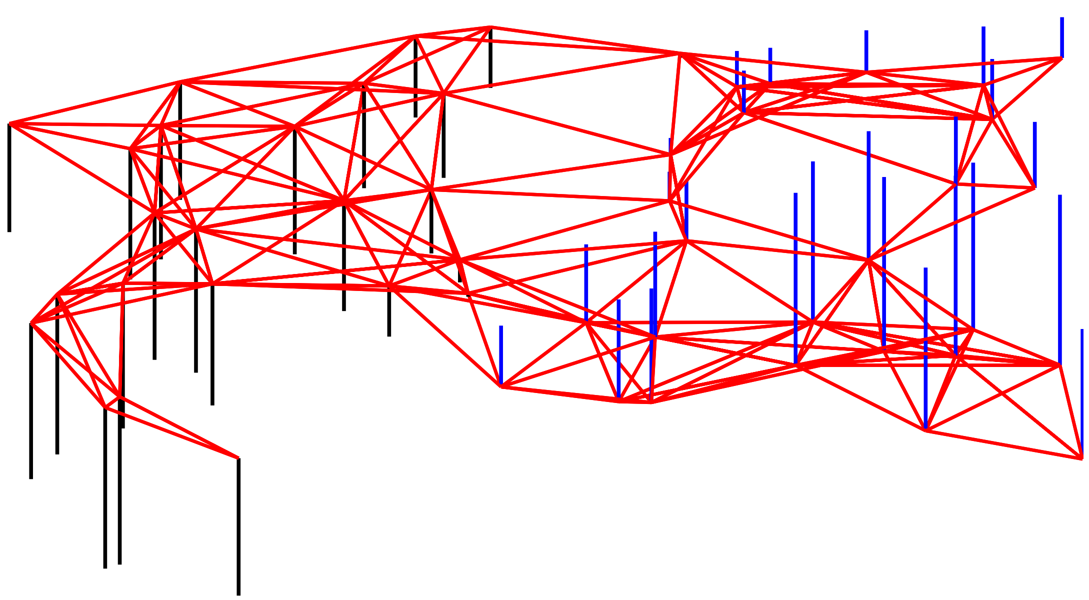
Agenda

- Graph Signal Processing Basics
  - Main definitions and operators
- Adaptive Graph Signal Representations
  - Graph Spectral Dictionaries
  - Dictionary Learning Algorithm
  - Applications of Graph Spectral Dictionaries
- Inferring Graphs from Observations
  - Factor Analysis Model
  - Graph Learning Algorithm
  - Illustrative Applications

Signals on Graphs

- Connected, undirected, weighted graph $\mathcal{G} = (V, E, W)$
  where $W_{i,j}$ is the weight of the edge $e = (i, j)$

- Graph signal: a function $f : \mathcal{V} \rightarrow \mathbb{R}$ that assigns real values to each vertex of the graph



- Graph description:
  - Degree matrix $\mathbf{D}$: diagonal matrix with sum of weights of incident edges
  - Laplacian matrix $\mathcal{L}$: difference operator defined based on $\mathbf{W}$

(Unormalized) Laplacian

- Laplacian is a difference operator $\mathcal{L} := \mathbf{D} - \mathbf{W}$

$$(\mathcal{L}f)(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}[f(i) - f(j)]$$

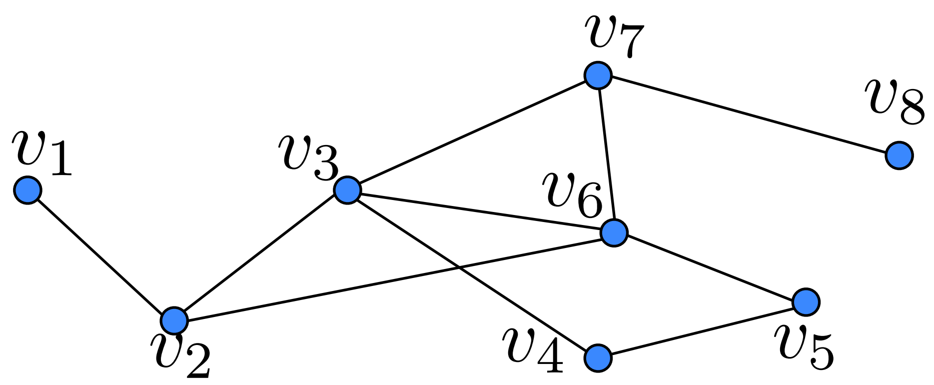
- It is a real symmetric matrix
- It has a complete set of eigenvectors $\{\mathbf{u}_\ell\}_{\ell=0,1,\ldots,N-1}$
- The eigenvectors are associated with real, nonnegative eigenvalues $\{\lambda_\ell\}_{\ell=0,1,\ldots,N-1}$

$$\mathcal{L}\mathbf{u}_\ell = \lambda_\ell \mathbf{u}_\ell, \ \forall \ell = 0, 1, \ldots, N-1$$

- Its spectrum is defined as $\sigma(\mathcal{L}) := \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_{N-1} := \lambda_{\max}$$

Laplacian example



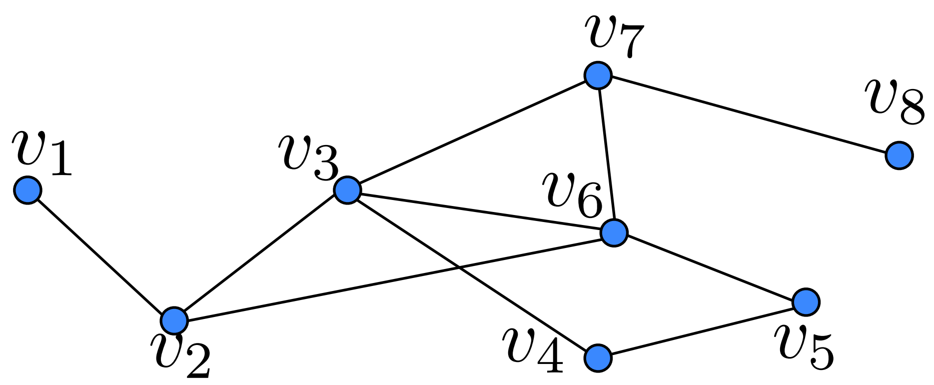
$$G = \{V, E\}$$

$$D = diag(degree(v_1) \quad \ldots \quad degree(v_n))$$

$$\mathcal{L} := \mathbf{D} - \mathbf{W}$$

$\mathbf{W}$ $\qquad\qquad\qquad\qquad$ $\mathcal{L}$

Laplacian example



$$G = \{V, E\}$$

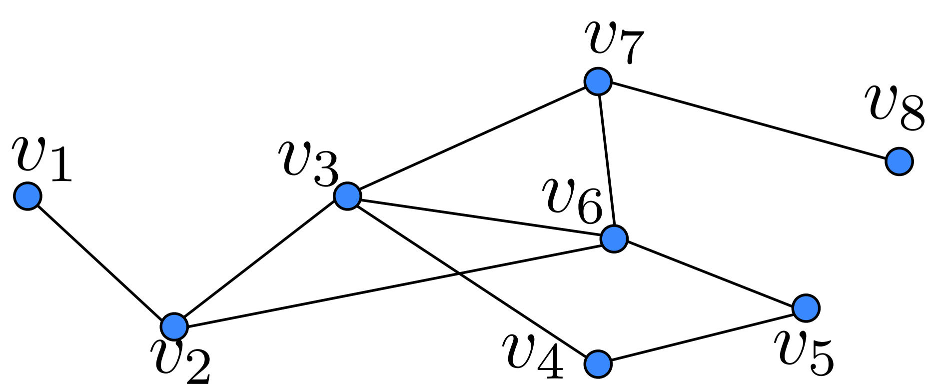
$$D = diag(degree(v_1) \quad ... \quad degree(v_n))$$

$$\mathcal{L} := \mathbf{D} - \mathbf{W}$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

$$\mathbf{W} \qquad\qquad\qquad \mathcal{L}$$

Laplacian example



$G = \{V, E\}$

$D = diag(degree(v_1) \quad \dots \quad degree(v_n))$

$\mathcal{L} := \mathbf{D} - \mathbf{W}$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

$\mathbf{W}$ $\qquad$ $\mathcal{L}$

- Symmetric
- Off-diagonal entries non-positive
- Rows sum up to zero
- Has a complete set of orthonormal eigenvectors: $L = \chi \Lambda \chi^T$

  $0 = \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{n-1}$

Normalized Laplacian

- The normalized Laplacian is another popular graph matrix
- Each weight $W_{i,j}$ is normalised by $\dfrac{1}{\sqrt{d_i d_j}}$

$$\tilde{\mathcal{L}} := \mathbf{D}^{-\frac{1}{2}} \mathcal{L} \mathbf{D}^{-\frac{1}{2}}$$

$$(\tilde{\mathcal{L}} f)(i) = \frac{1}{\sqrt{d_i}} \sum_{j \in \mathcal{N}_i} W_{i,j} \left[\frac{f(i)}{\sqrt{d_i}} - \frac{f(j)}{\sqrt{d_j}} \right]$$

- The set of eigenvalues is $0 = \tilde{\lambda}_0 < \tilde{\lambda}_1 \leq \ldots \leq \tilde{\lambda}_{\max} \leq 2$
- The normalized Laplacian has often stability benefits

Graph Fourier Transform

- The eigenvectors of the graph Laplacian are used for defining the Graph Fourier Transform

GFT

$$\hat{f}(\lambda_\ell) := \langle \mathbf{f}, \mathbf{u}_\ell \rangle = \sum_{i=1}^{N} f(i) u_\ell^*(i)$$

IGFT

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell) u_\ell(i)$$

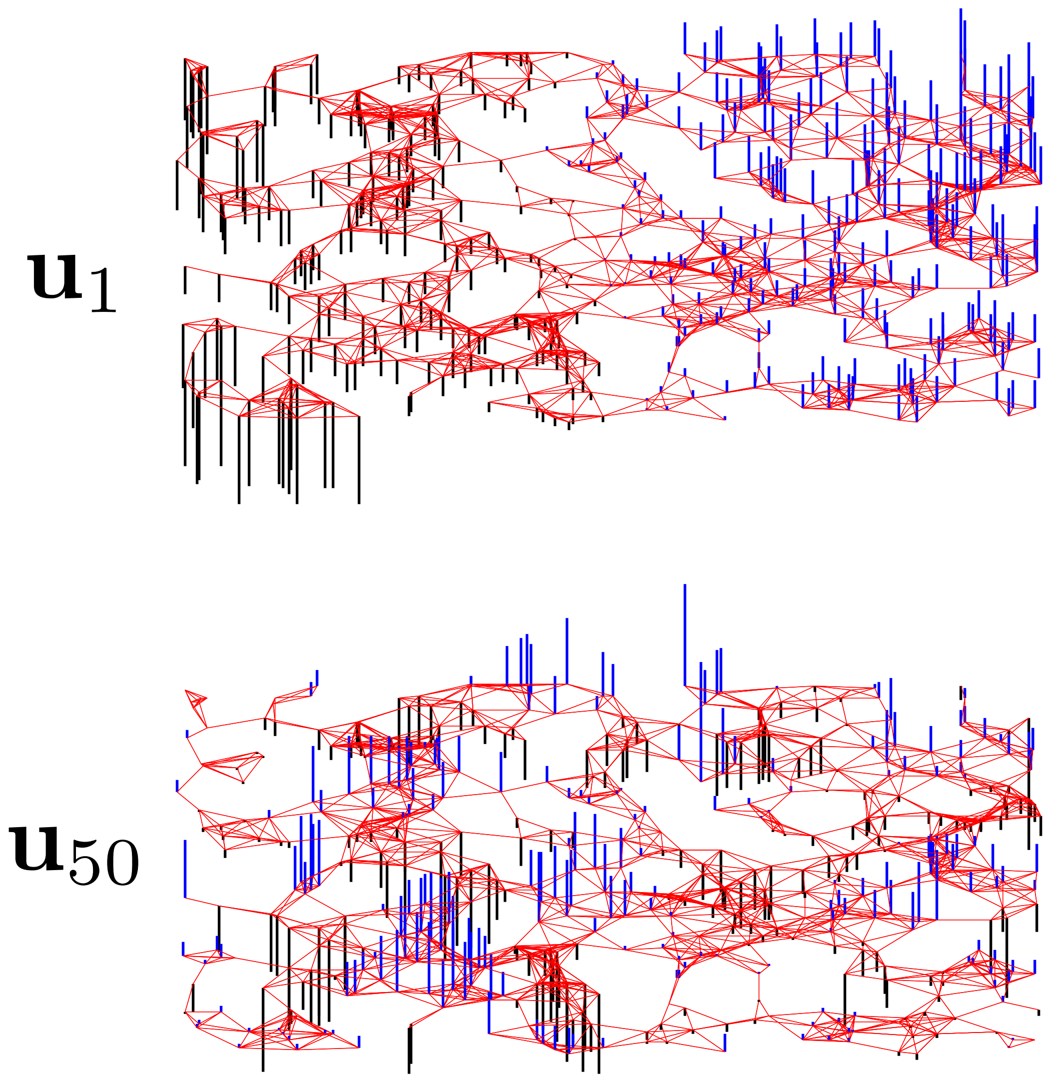
- This is analogous to the classical Fourier Transform built on eigenfunctions of the 1-D Laplace operator

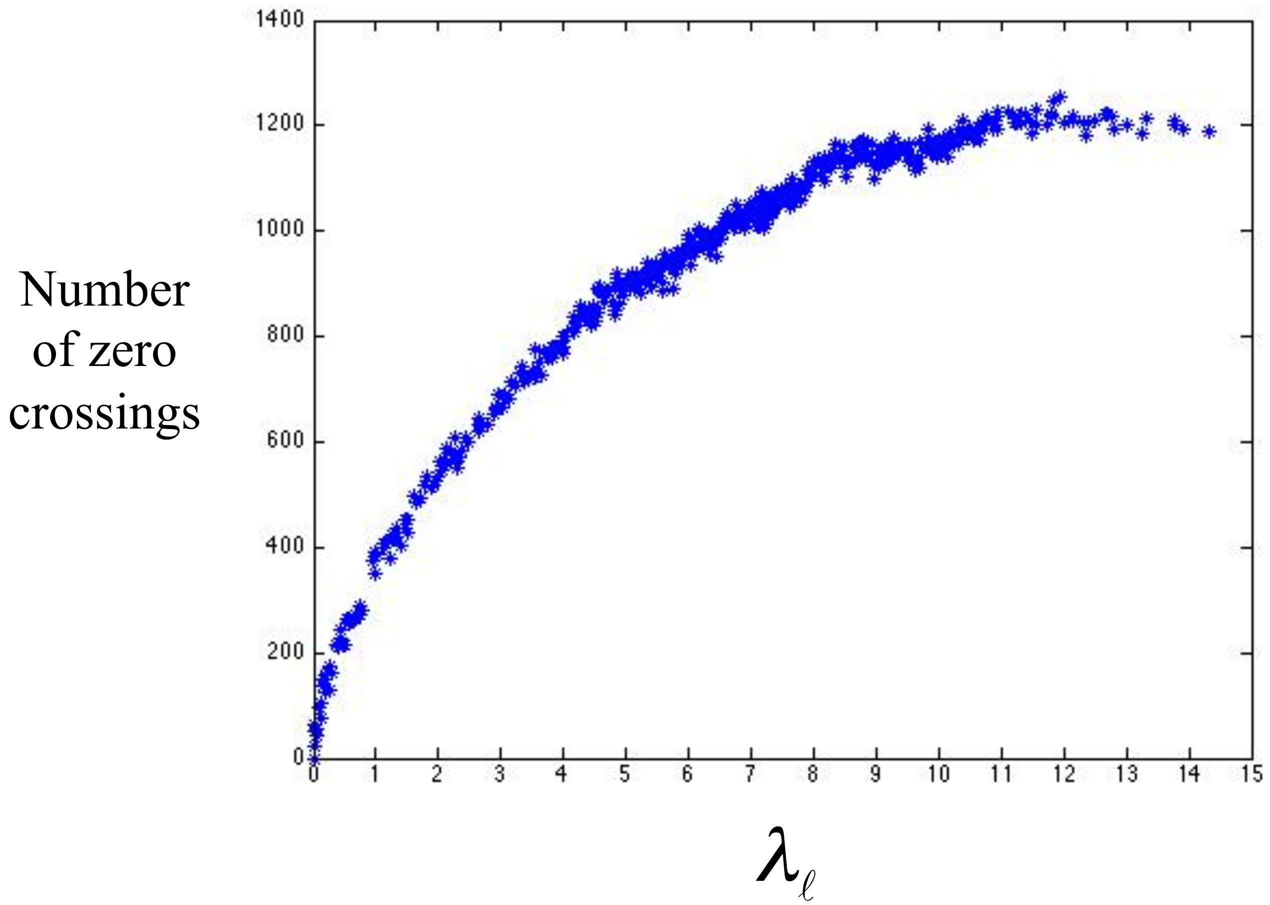
$$\hat{f}(\xi) := \langle f, e^{2\pi i \xi t} \rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

$$-\Delta(e^{2\pi i \xi t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t}$$

Notion of 'frequency'

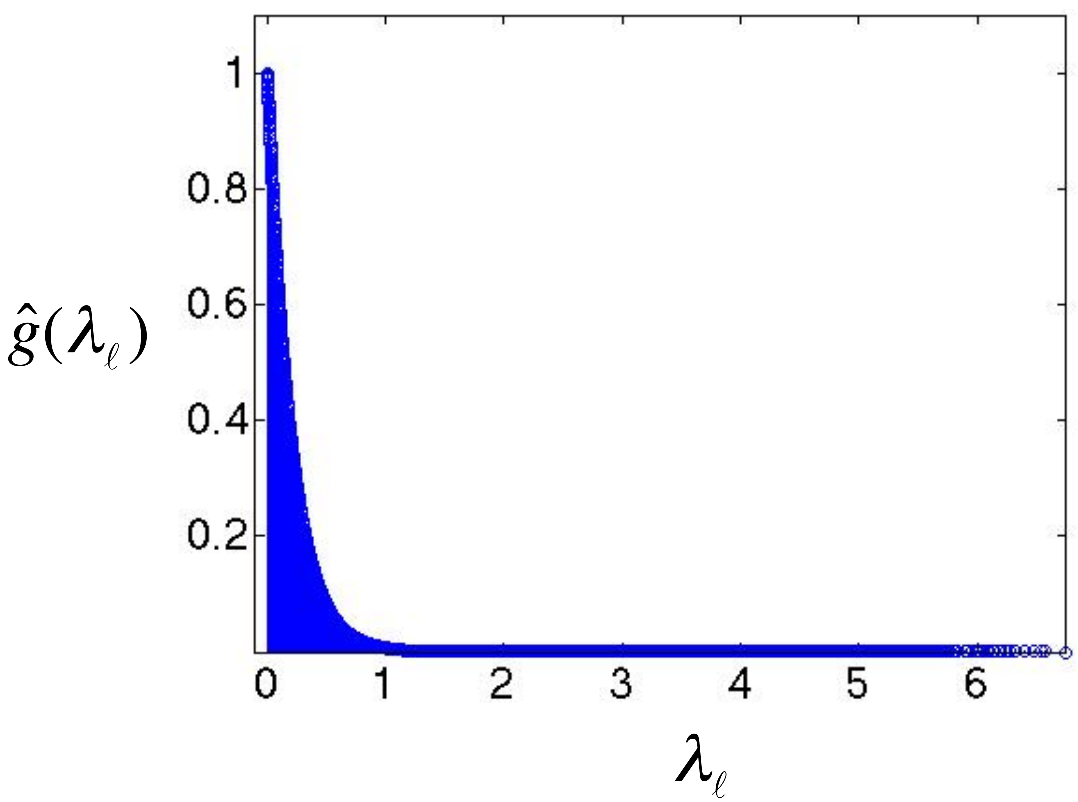
- The graph Laplacian eigenvalues and eigenvectors carry a notion of frequency



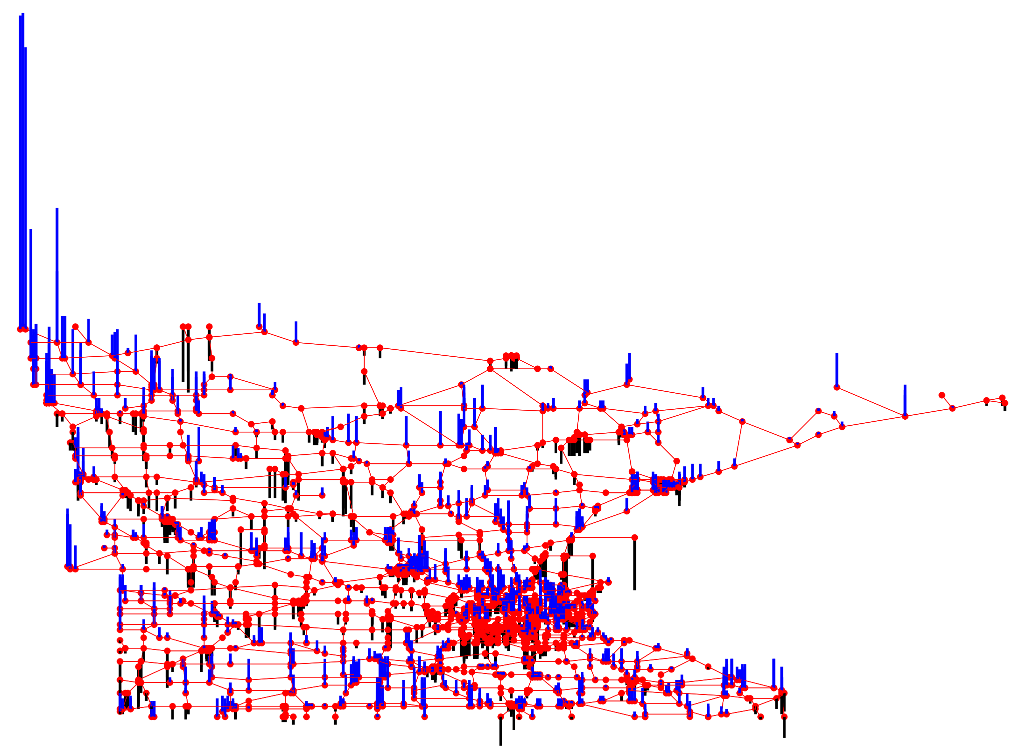


Dual representations

- Graph signals represented in either the vertex or the spectral domains (*kernels, or graph Fourier multipliers*)



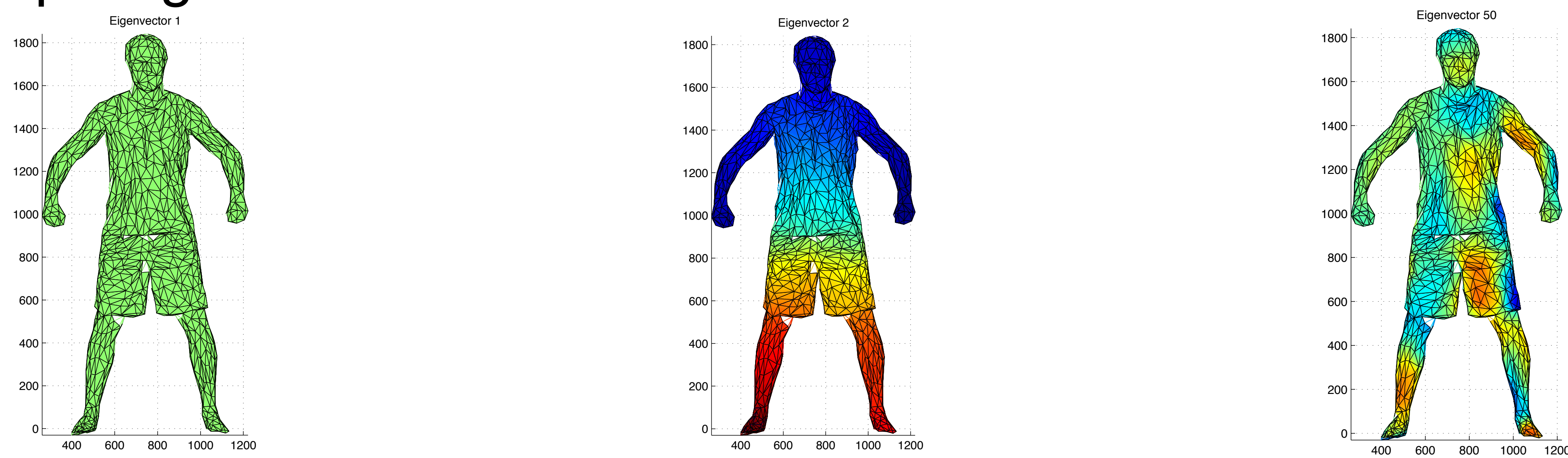
$$\hat{g}(\lambda_\ell) = e^{-5\lambda_\ell}$$



$$g(n) \xleftarrow{IGFT} \hat{g}(\lambda_\ell)$$

One last GFT illustration

- The eigenvectors of the Laplacian provide a harmonic analysis of graph signals



Low frequency → High frequency

GFT:

$$\hat{y}\left(\lambda_{\ell}\right) = \langle y, \chi_{\ell} \rangle = \sum_{n=1}^{N} y(n)\chi_{\ell}^{*}(n), \quad \ell = 0, 1, ..., N-1$$

IGFT:

$$y(n) = \sum_{\ell=0}^{N-1} \hat{y}\left(\lambda_{\ell}\right) \chi_{\ell}(n), \quad \forall n \in \mathcal{V}$$

- They have global support on the graph!

Frequency filtering

- Analogously to classical filtering, one can perform graph spectral filtering with transfer function $\hat{h}(\lambda_\ell)$

$$\hat{f}_{out}(\lambda_\ell) = \hat{f}_{in}(\lambda_\ell)\hat{h}(\lambda_\ell)$$

- Equivalently $$f_{out}(i) = \sum_{\ell=0}^{N-1} \hat{f}_{in}(\lambda_\ell)\hat{h}(\lambda_\ell)u_\ell(i)$$

- In matrix notation: $$\mathbf{f}_{out} = \hat{h}(\mathcal{L})\mathbf{f}_{in}$$

$$\hat{h}(\mathcal{L}) := \mathbf{U} \begin{bmatrix} \hat{h}(\lambda_0) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \hat{h}(\lambda_{N-1}) \end{bmatrix} \mathbf{U}^{\mathrm{T}}$$

Filtering in the vertex domain

- Linear combination of values at neighbour vertices

$$f_{out}(i) = b_{i,i} f_{in}(i) + \sum_{j \in \mathcal{N}(i,K)} b_{i,j} f_{in}(j)$$

  - localized linear transform

- Example: polynomial filter as $\hat{h}(\lambda_\ell) = \sum_{k=0}^{K} a_k \lambda_\ell^k$

$$f_{out}(i) = \sum_{\ell=0}^{N-1} \hat{f}_{in}(\lambda_\ell) \hat{h}(\lambda_\ell) u_\ell(i)$$

$$= \sum_{j=1}^{N} f_{in}(j) \sum_{k=0}^{K} a_k \left(\mathcal{L}^k\right)_{i,j} \rightarrow b_{i,j} := \sum_{k=d_{\mathcal{G}}(i,j)}^{K} a_k \left(\mathcal{L}^k\right)_{i,j}$$

Translation on graphs

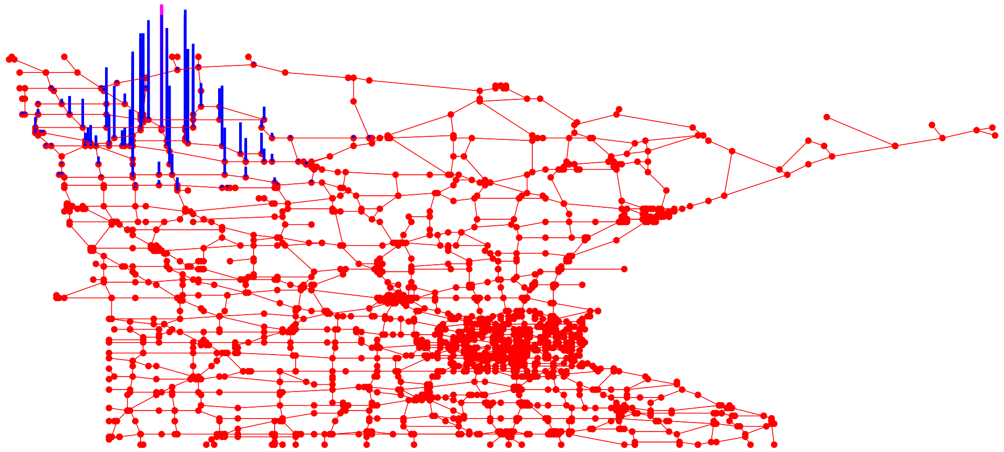
- The classical translation $(T_u f)(t) := f(t-u)$ does not generalise to non-regular graphs
- A generalized translation operator on graphs can still be defined as

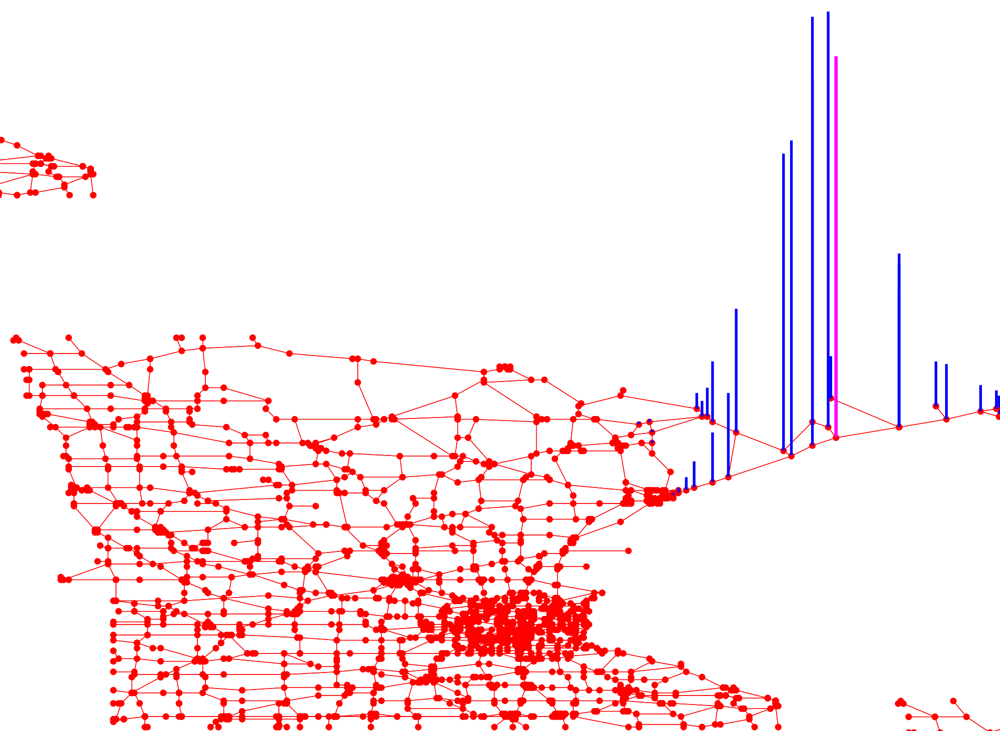
$$T_n : \mathbb{R}^N \to \mathbb{R}^N$$

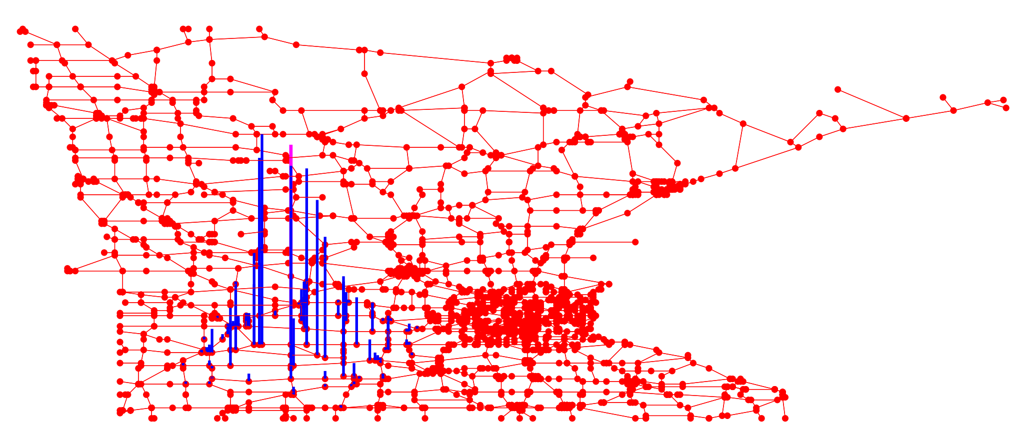
$$\boxed{(T_n g)(i) := \sqrt{N}(g * \delta_n)(i) = \sqrt{N} \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) u_\ell^*(n) u_\ell(i)}$$

$$\delta_n(i) = \begin{cases} 1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \qquad (f * h)(i) := \sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell)\hat{h}(\lambda_\ell) u_\ell(i)$$

Translation example







Translation of the heat kernel to different locations of the Minnesota graph

Transforms on graphs

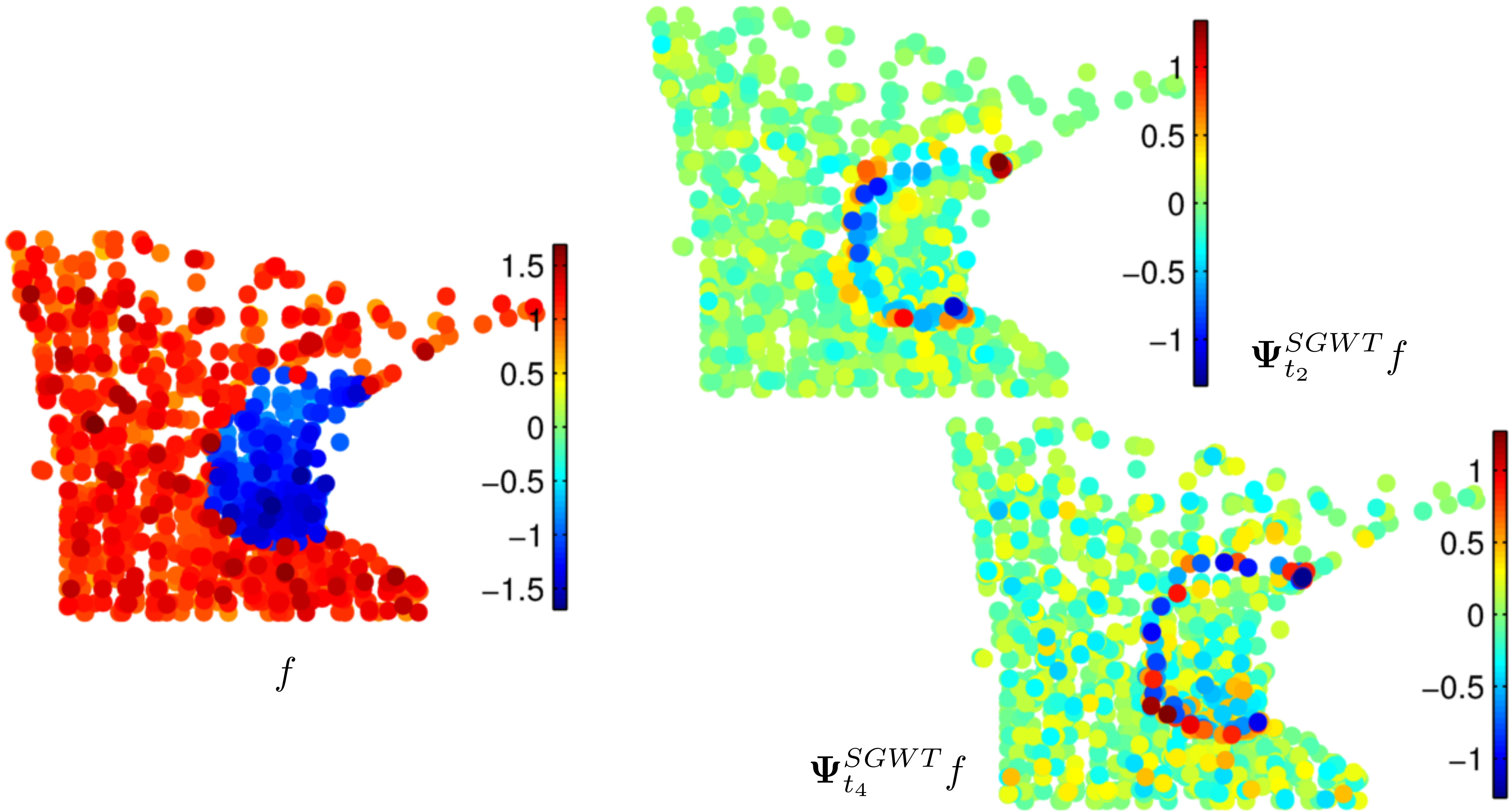
- Localized transforms are ideal to analyse graph signals
  - analysis properties and scalable implementations
  - GFT is unfortunately not a local transform
- Wavelet transforms are particularly interesting
  - localization in both the vertex and spectral domains
  - different designs in the vertex or the spectral domain [Shuman:2013]
  - *example*: Spectral Graph Wavelets [Hammond:2011]

$$\mathbf{\Psi}^{SGWT} : \mathbb{R}^N \rightarrow \mathbb{R}^{N(K+1)} \qquad \mathbf{\Psi}^{SGWT} = [\mathbf{\Psi}^{SGWT}_{scal}; \mathbf{\Psi}^{SGWT}_{t_1}; \ldots; \mathbf{\Psi}^{SGWT}_{t_K}]$$

  - Dilations and translations of a band-pass kernel $\psi^{SGWT}_{t_k,i} := T_i \mathcal{D}_{t_k} \mathbf{g} = \widehat{\mathcal{D}_{t_k} g}(\mathcal{L}) \boldsymbol{\delta}_i$
  - Translation of a low-pass kernel $\psi^{SGWT}_{scal,i} := T_i \mathbf{h} = \hat{h}(\mathcal{L}) \boldsymbol{\delta}_i$

- Such transforms are dependent on graph (not signal)

SGWT illustration



[Shuman:2013]

Data-adaptive representations

- Sparse graph signal representation

- We want to have an efficient structured representation that is adapted to data: *graph spectral dictionaries*

Data-adaptive representations

- Sparse graph signal representation

$$\boxed{\mathbf{y} = \mathbf{\Phi x} \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leq T_0}$$

- We want to have an efficient structured representation that is adapted to data: *graph spectral dictionaries*

Data-adaptive representations

- Sparse graph signal representation

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leq T_0$$

structure-based methods

$\mathbf{\Phi}$ defined
via mathematical structures
(*transforms*):
Fourier, wavelets...

- We want to have an efficient structured representation that is adapted to data: *graph spectral dictionaries*

Data-adaptive representations

- Sparse graph signal representation

$$\mathbf{y} = \mathbf{\Phi x} \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leq T_0$$

structure-based methods

$\mathbf{\Phi}$ defined
via mathematical structures
(*transforms*):
Fourier, wavelets...

numerical methods

$\mathbf{\Phi}$ learned
via training samples
(*dictionary learning*):
sparse coding, MOD, K-SVD...

- We want to have an efficient structured representation that is adapted to data: *graph spectral dictionaries*

Data-adaptive representations

- Sparse graph signal representation

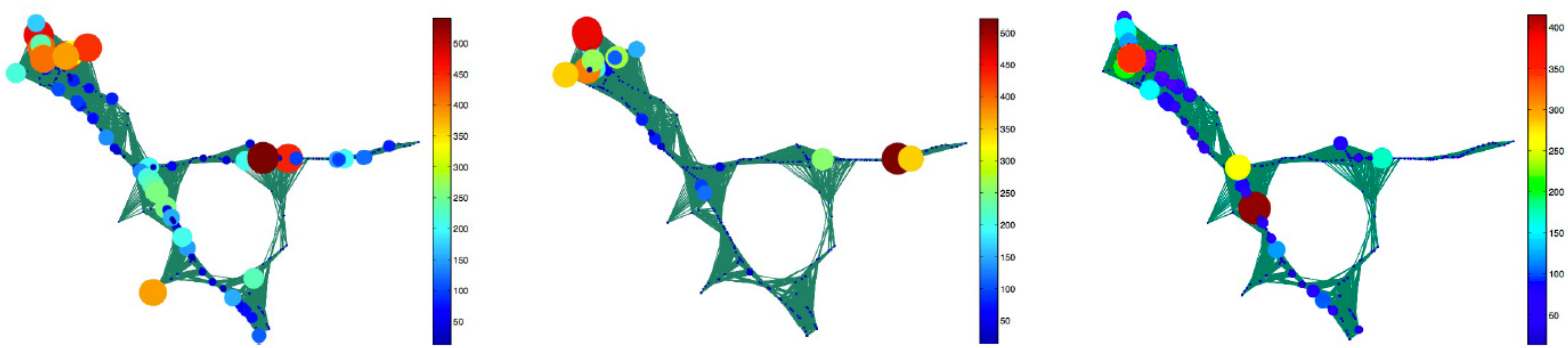
$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leq T_0$$

structure-based methods	numerical methods
$\mathbf{\Phi}$ defined via mathematical structures (*transforms*): Fourier, wavelets...	$\mathbf{\Phi}$ learned via training samples (*dictionary learning*): sparse coding, MOD, K-SVD...
not data-specific!	**not graph-specific!**

- We want to have an efficient structured representation $\mathbf{\Phi}$ that is adapted to data: *graph spectral dictionaries*

Dictionary for Graph Signals

- Our objective: to learn meaningful graph signal representations that
  - ✓ reveal relevant structural properties of the graph signals/extract important features on graphs
  - ✓ sparsely represent different classes of signals on graphs

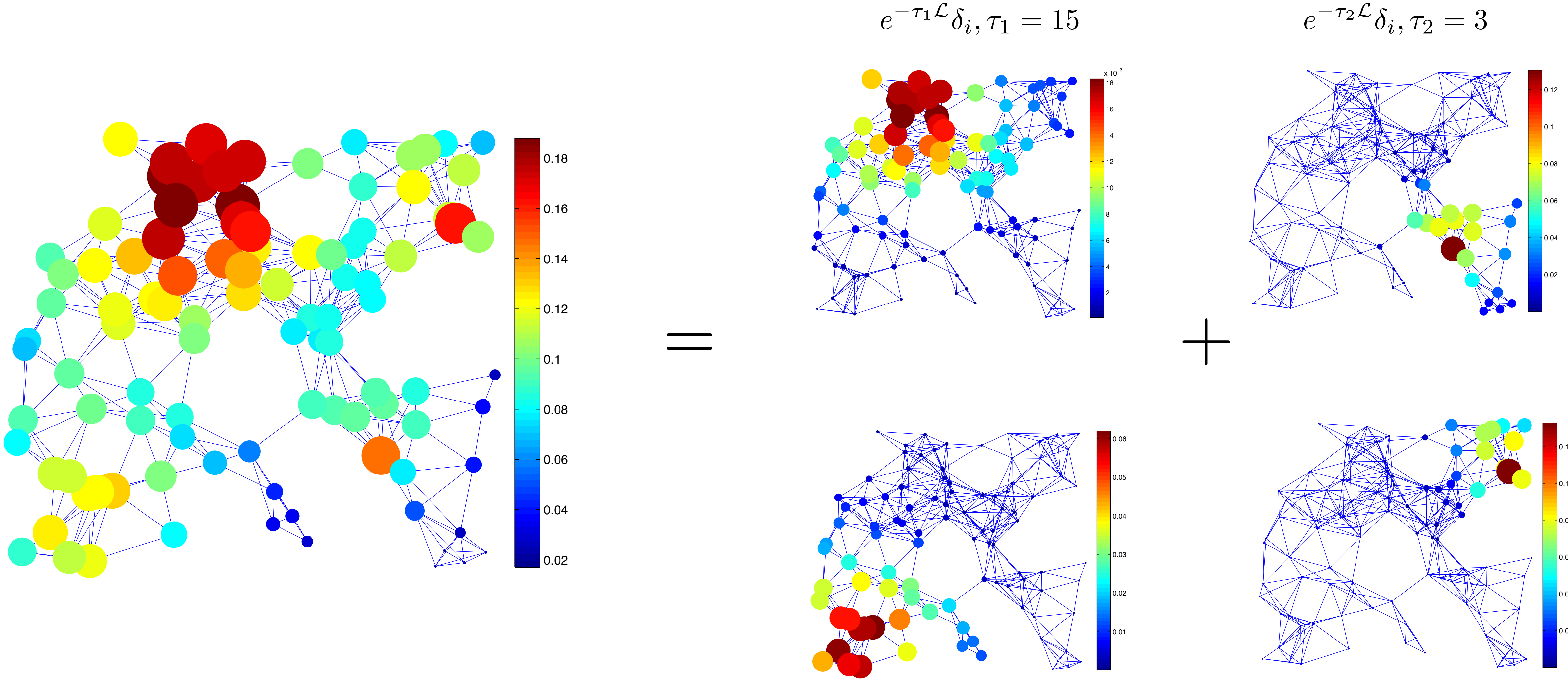


(a) Day 1 (b) Day 2 (c) Day 3

How can we define atoms on graphs?

Sparse signal model

- Graph signals can be approximated by a small number of localized components
  - e.g., multiple processes started at different vertices



Parametric graph atoms

- A set of generating kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,\ldots,S}$ represent the spectral characteristics of the signals
- The kernels are chosen to be smooth polynomial of degree K in order to form localized graph features

$$\hat{g}(\lambda_\ell) = \sum_{k=0}^{K} \alpha_k \lambda_\ell^k, \quad \ell = 0, \ldots, N-1$$

- A graph atom is the translation of the kernel to vertex *n*

$$\boxed{T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{\ell=0}^{N-1} \sum_{k=0}^{K} \alpha_k \lambda_\ell^k \chi_\ell^*(n) \chi_\ell = \sqrt{N} \sum_{k=0}^{K} \alpha_k (\mathcal{L}^k)_n}$$

$\mathcal{L}$: normalized Laplacian, χ_ℓ : eigenvector

Dictionary Structure

- A parametric graph dictionary $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_S]$ is a concatenation of S subdictionaries

- Each subdictionary is built on a specific kernel

$$\mathcal{D}_s = \widehat{g}_s(\mathcal{L}) = \chi \left(\sum_{k=0}^{K} \alpha_{sk} \Lambda^k \right) \chi^T = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k$$

  - Each atom (column of $\mathcal{D}_s$) corresponds to a K-hop localized pattern centered on a node of the graph, i.e.,

$$\frac{1}{\sqrt{N}} T_n g_s$$

Dictionary Learning Problem

- Learning consists in computing $\{\alpha_{sk}\}_{s=1,2,\ldots,S;\ k=1,2,\ldots,K}$
- Given a set of training signals $Y = [y_1, y_2, ..., y_M] \in \mathbb{R}^{N \times M}$ on the graph $\mathcal{G}$, solve

$$
\begin{aligned}
\underset{\alpha \in \mathbb{R}^{(K+1)S},\ X \in \mathbb{R}^{SN \times M}}{\operatorname{argmin}} \quad & \{||Y - \mathcal{D}X||_F^2 + \mu\|\alpha\|_2^2\} \\
\text{subject to} \quad & \|x_m\|_0 \le T_0, \quad \forall m \in \{1, ..., M\}, \\
& \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \quad \forall s \in \{1, 2, ..., S\} \\
& 0 \preceq \mathcal{D}_s \preceq c, \quad \forall s \in \{1, 2, ..., S\} \\
& (c - \epsilon_1) I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2) I,
\end{aligned}
$$

The spectral constraints guarantee that:
1. The learned kernels cover the whole spectrum
2. The dictionary is a frame

Alternating optimisation

Algorithm 1 Parametric Dictionary Learning on Graphs

Input: Signal set Y, initial dictionary $\mathcal{D}^{(0)}$, target signal sparsity T_0, polynomial degree K, number of subdictionaries S, number of iterations $iter$
Output: Sparse signal representations X, polynomial coefficients α
Initialization: $\mathcal{D} = \mathcal{D}^{(0)}$
for $i = 1, 2, ..., iter$ **do:**
    Sparse Approximation Step:
        (a) Scale each atom in $\mathcal{D}$ to a unit norm
        (b) Update X using Sparse Coding
        (c) Rescale X, $\mathcal{D}$ to recover the polynomial structure
    Dictionary Update Step:
        Compute the polynomial coefficients α and update the dictionary
end for

Sparse Coding Step

- The dictionary (α) is fixed
- The sparse coding coefficients are computed with

$$\underset{X}{\operatorname{argmin}} ||Y - \mathcal{D}X||_F^2 \text{ subject to } \|x_m\|_0 \leq T_0$$

$$\forall m \in \{1, ..., M\}$$

- this can be solved by greedy algorithmms, like OMP
- it can also be solved by convex relaxation using iterative soft thresholding, for example

Dictionary Update Step

- The coefficients *X* are fixed, the dictionary is updated with

$$\underset{\alpha \in \mathbb{R}^{(K+1)S}}{\operatorname{argmin}} \left\{ ||Y - \mathcal{D}X||_F^2 + \mu \|\alpha\|_2^2 \right\}$$

$$\text{subject to} \quad \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \quad \forall s \in \{1, 2, ..., S\}$$

$$0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, ..., S\}$$

$$(c - \epsilon_1)I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2)I.$$

  - quadratic function with affine constraints, solved by interior point methods or ADMM

Properties of the dictionary

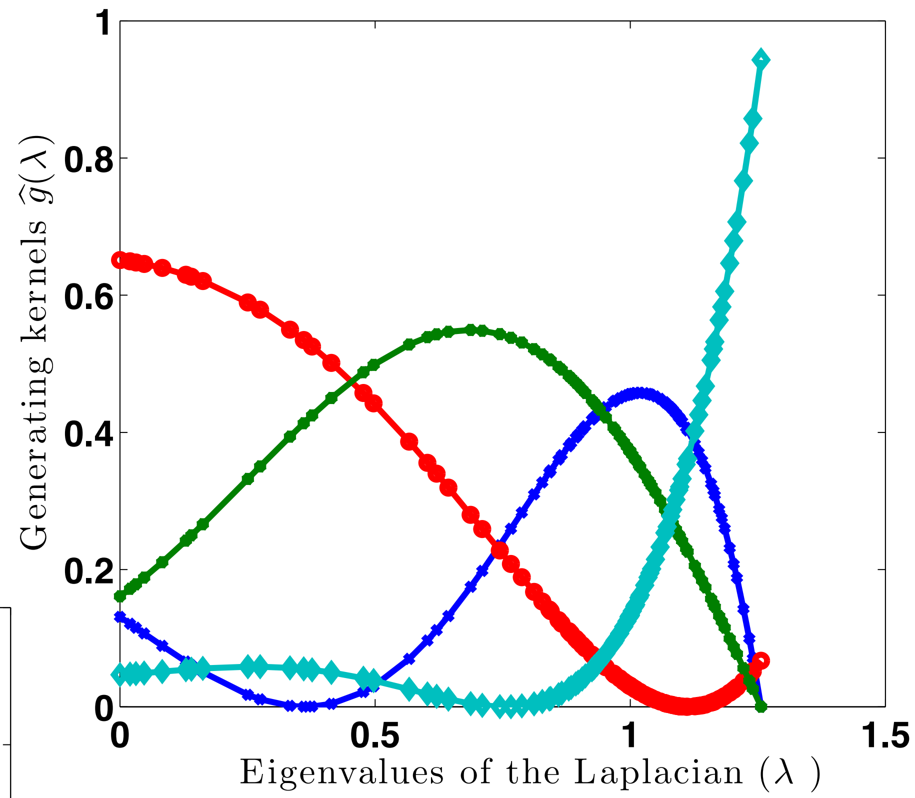
- By construction, the dictionary is a frame
- The coherence depends on the graph

$$\phi \leq \max_{n \neq n', s, s'} \frac{\nu(\sum_{\ell=0}^{N-1} |\widehat{g_s}(\ell)\widehat{g_{s'}}(\ell)|^2)^{1/2} \|deg\|^2}{|\widehat{g_s}(\lambda_0)||\widehat{g'_s}(\lambda_0)|\sqrt{deg_n}\sqrt{deg_{n'}}}$$

- Parametric structure: easy dictionary description
  - it has only $(K+1)S$ parameters
- Polynomial form: efficient implementation, esp. when the graph is sparse
  - Both forward and adjoint operators can be efficiently applied

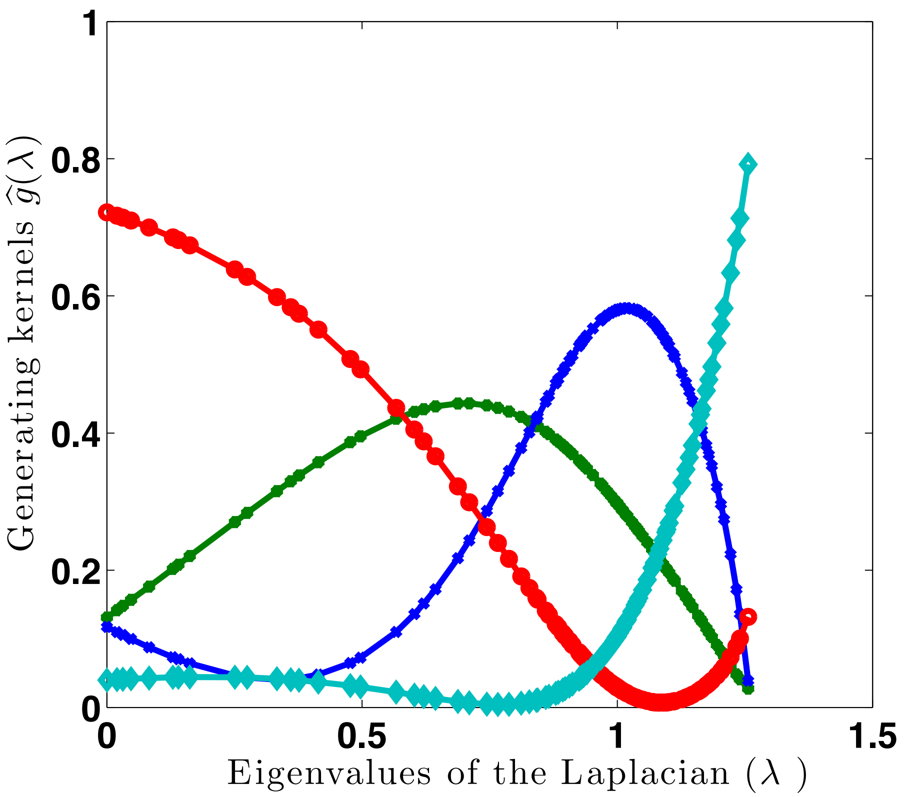
$$\mathcal{D}^T y = \sum_{s=1}^{S} \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k y \qquad \mathcal{D}\mathcal{D}^T y = \sum_{s=1}^{S} \widehat{g_s}^2(\mathcal{L}) y$$

Recovery on synthetic data

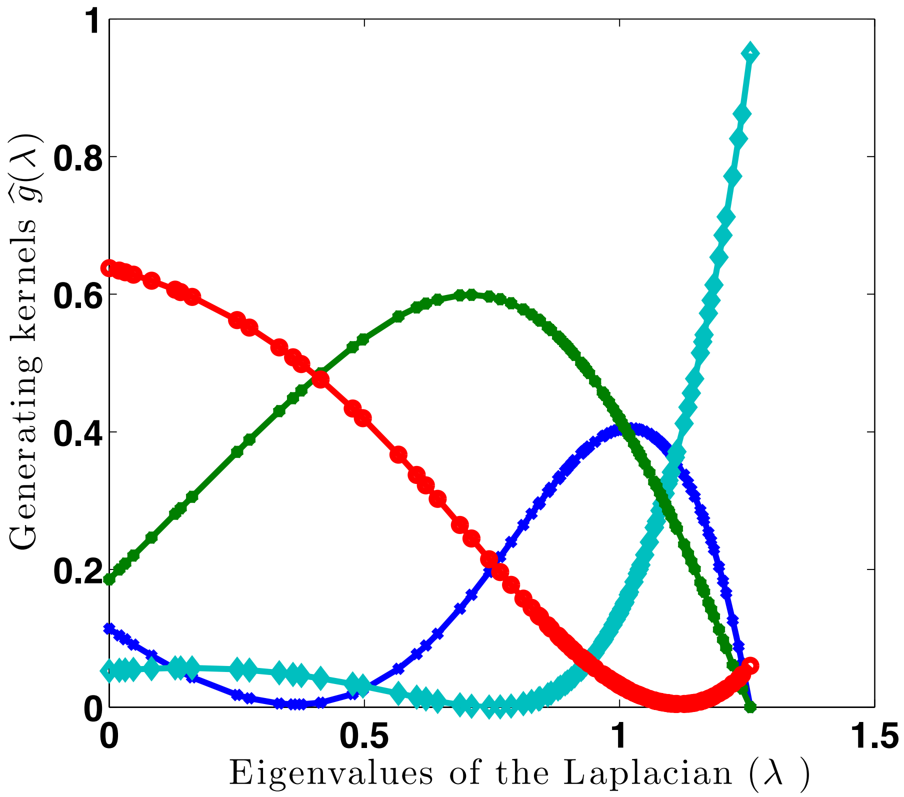


Original polynomial kernels

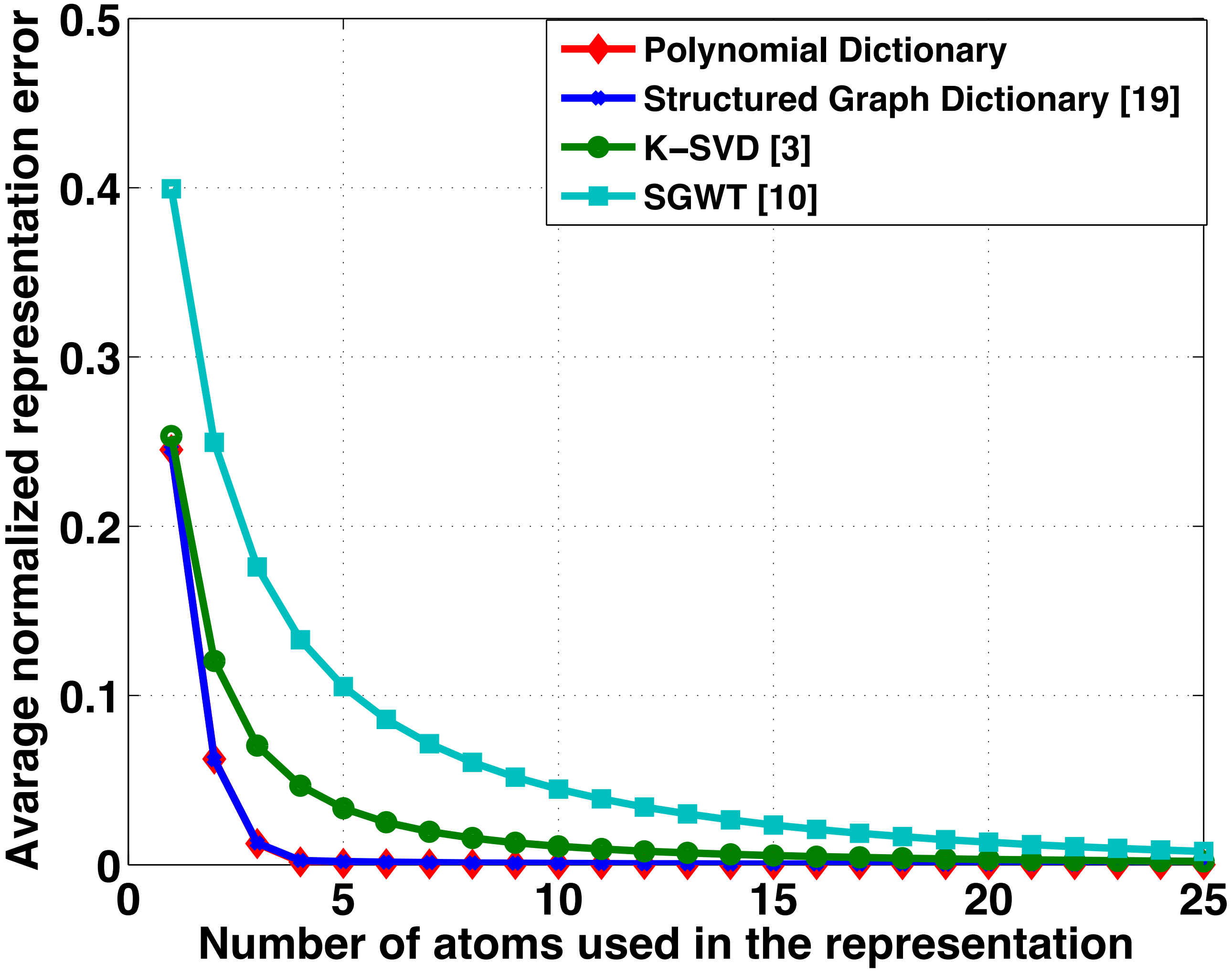
Learned kernels (M=400)



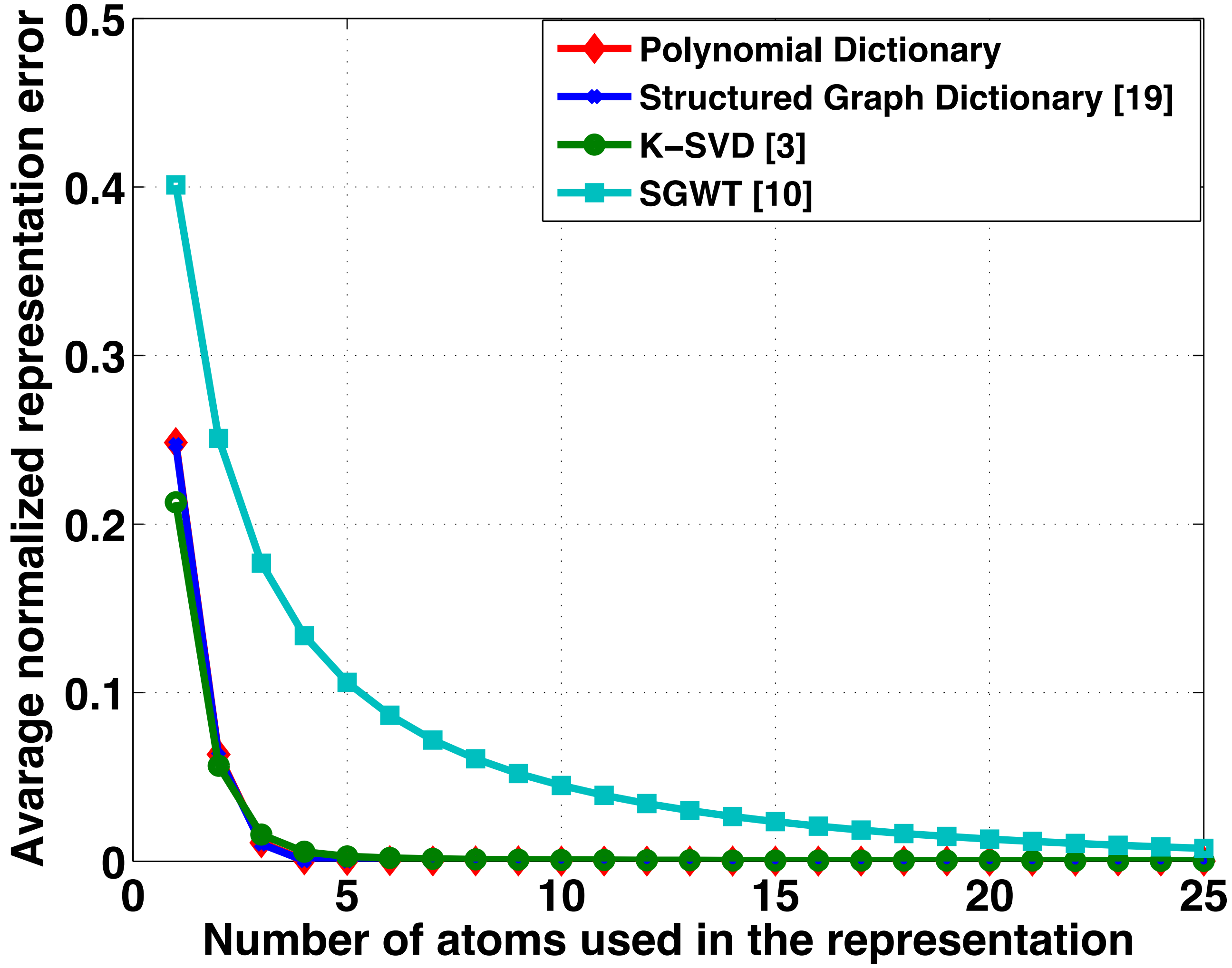
Learned kernels (M=2000)



Approximation on synthetic data



M=400

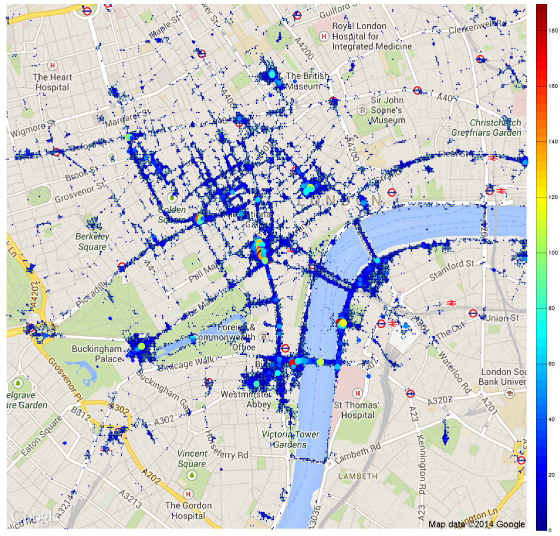


M=2000

Real World Datasets

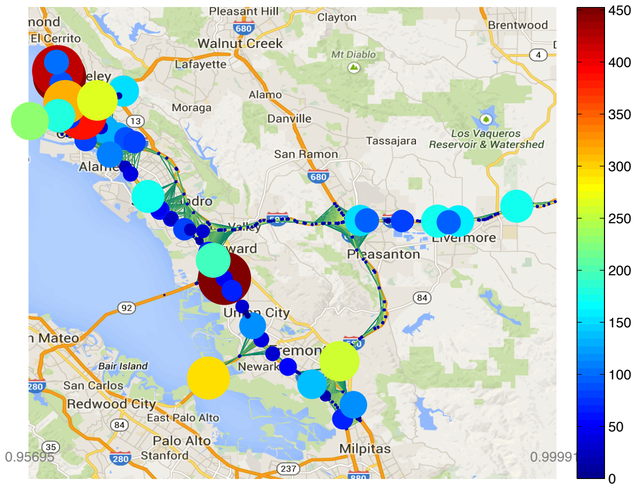
Flickr

- **Nodes:** 245 vertices in the Trafalgar Square (London), each representing a geographical area
- **Edges**: Assign edge when distance < 30m
- **Graph signals:** Daily number of distinct users that took photos between Jan. 2010 and June 2012



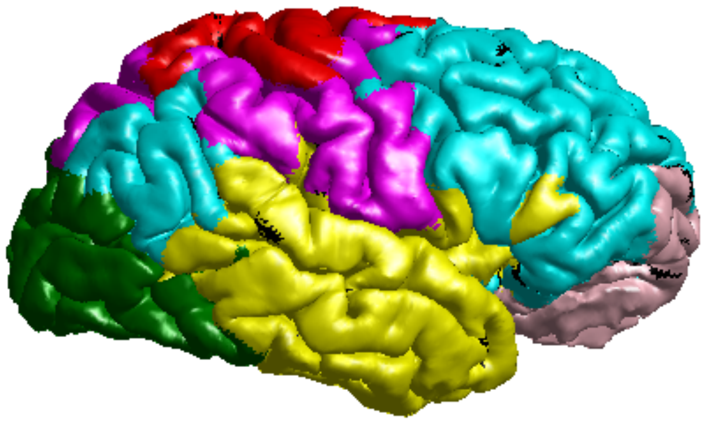
Traffic

- **Nodes:** 439 detector stations in Alameda County, CA
- **Edges**: Assign edge when distance < 13km
- **Graph signals:** Daily number of bottlenecks (in minutes) between Jan. 2007 to May. 2013

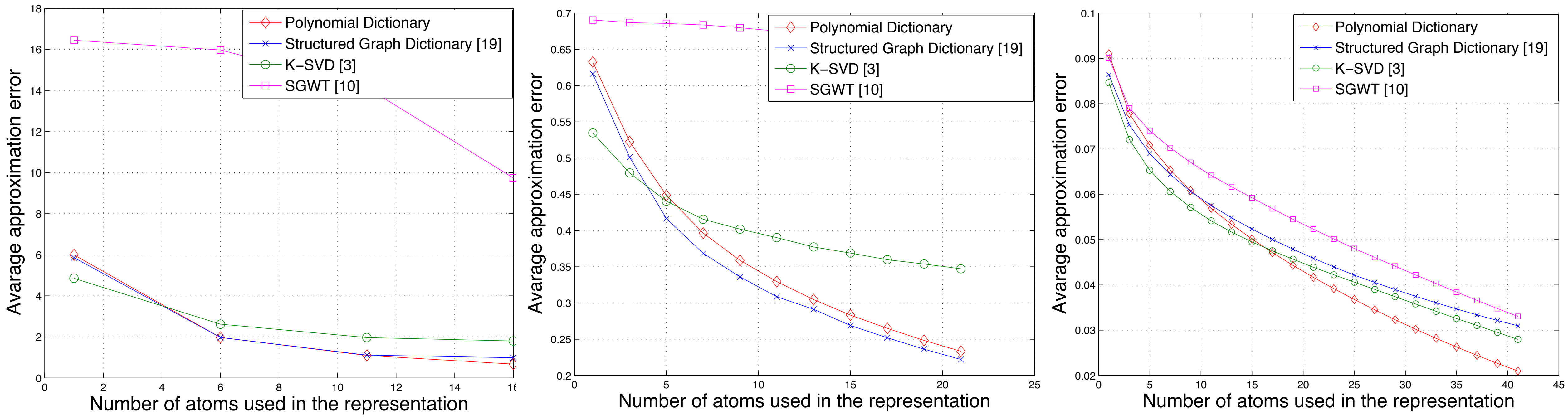


Brain

- **Nodes:** 88 brain regions of contiguous voxels
- **Edges**: Assign edge if anatomical distance < 40 mm
- **Graph signals:** fMRI signals acquired on five subjects, in different states, 1290 signals per subject



Approximation performance

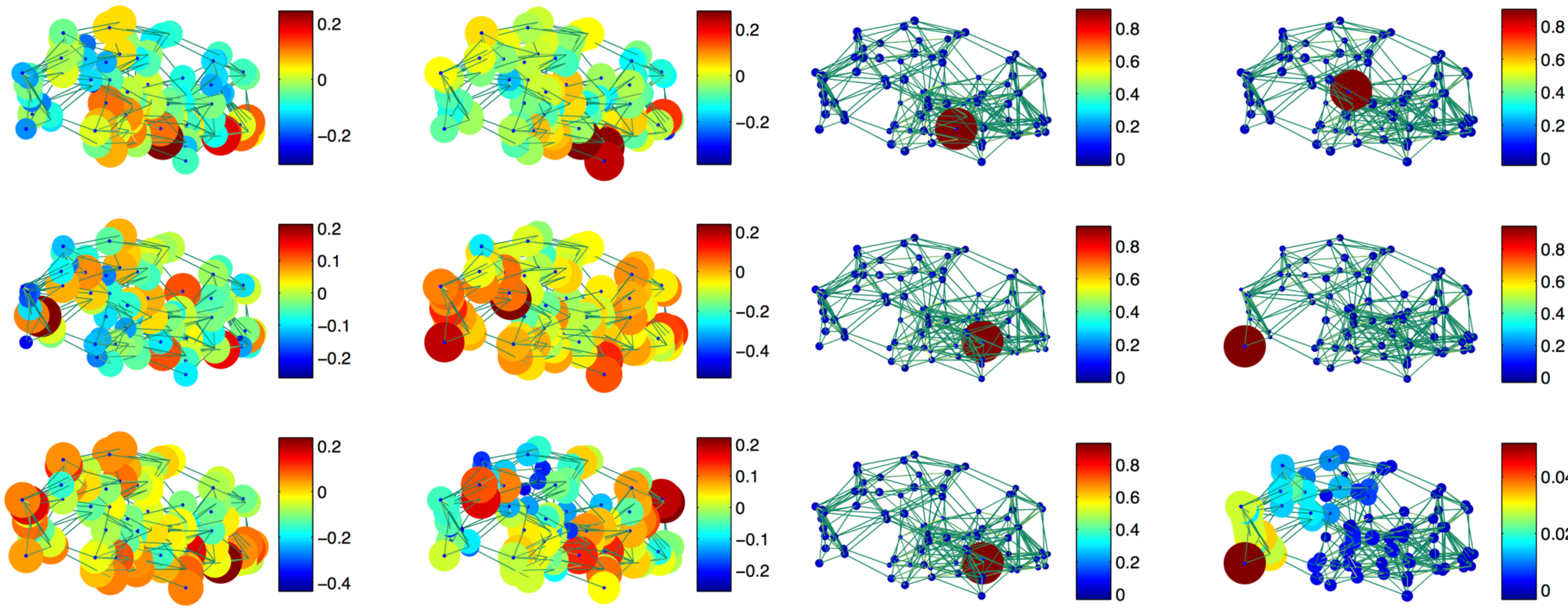


(a) Flickr dataset **(b) Traffic dataset** **(c) Brain dataset**

- As the sparsity level increases, the localisation property becomes beneficial
- The polynomial dictionary is able to learn local patterns in areas of the graph that do not show up in the training signals

Examples of Learned Atoms

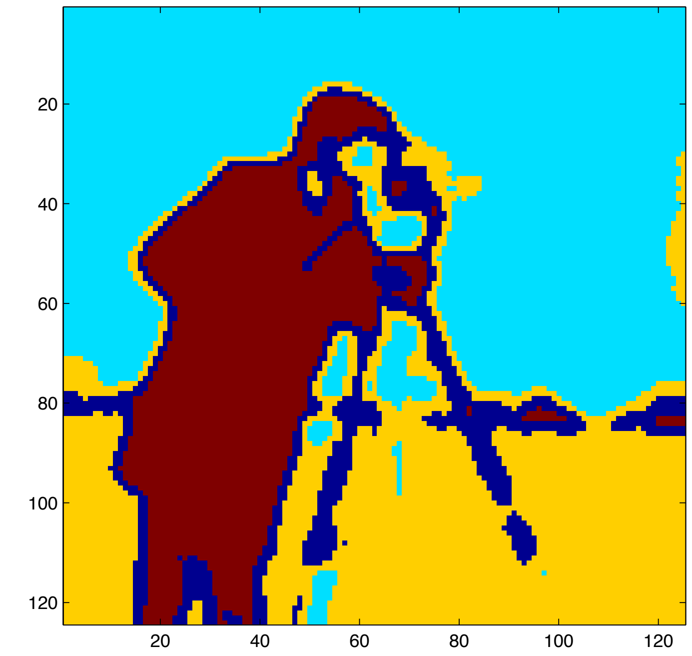
- Most common atoms in OMP expansions

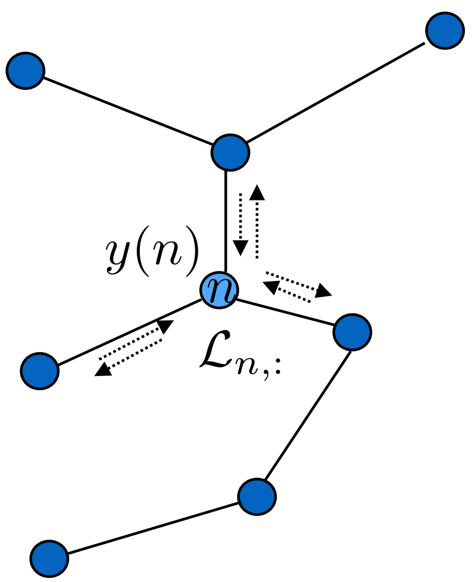


K-SVD Dictionary **Polynomial Graph Dictionary**

Applications of graph dictionaries

- Graph dictionaries apply to many sparse problems
  - helpful when smooth priors are insufficient
- Graph dictionaries also define features on graphs
  - learning or clustering applications
- By construction, spectral graph dictionaries lead to effective implementations
  - distributed processing applications in networks



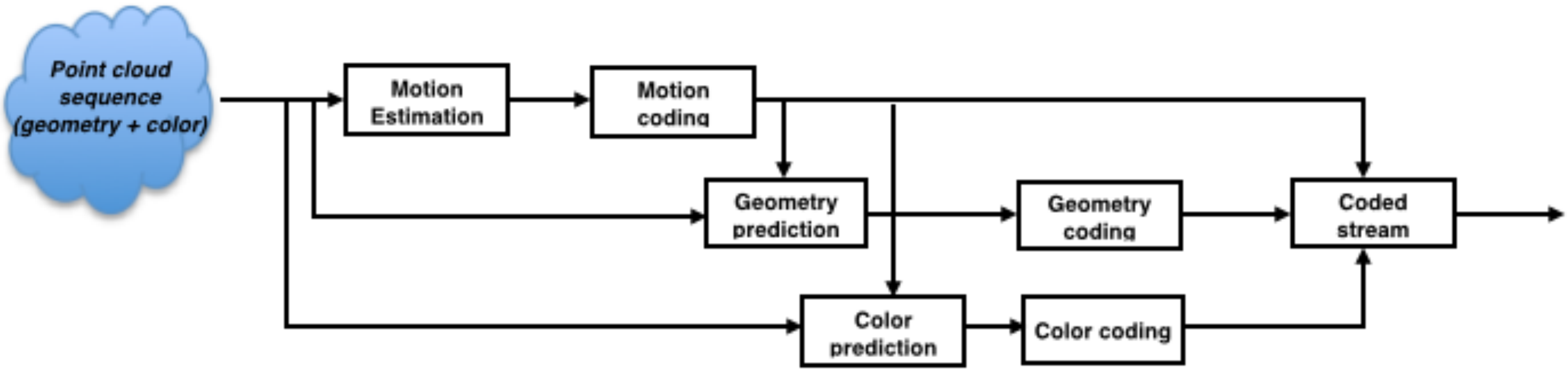


3D Point Cloud Sequences

- No explicit spatio-temporal geometry structure
  - Frames have different number of points
  - No association between points over time

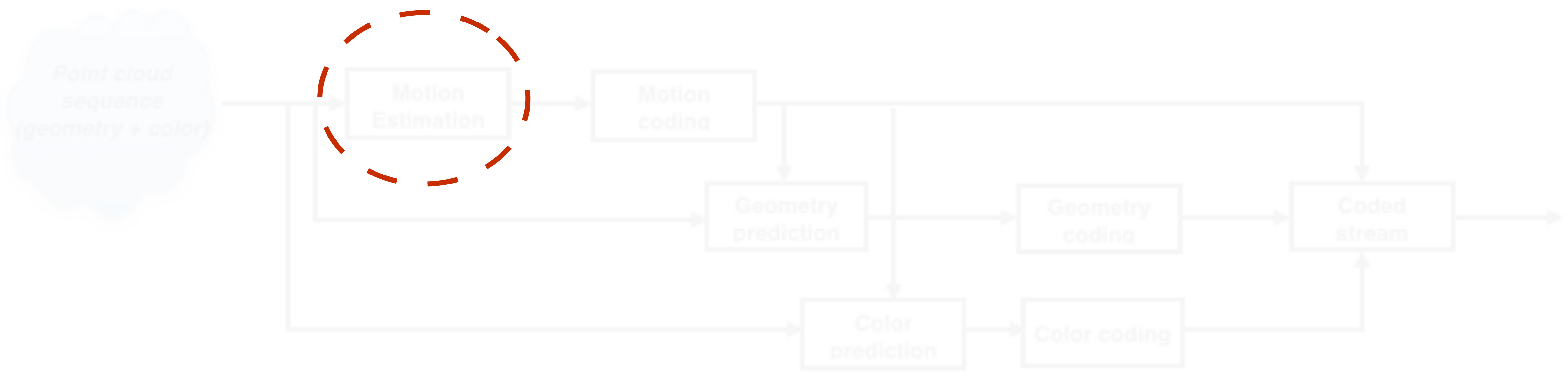
- Graph localised features can be used to match frames

Graph-based Motion Estimation



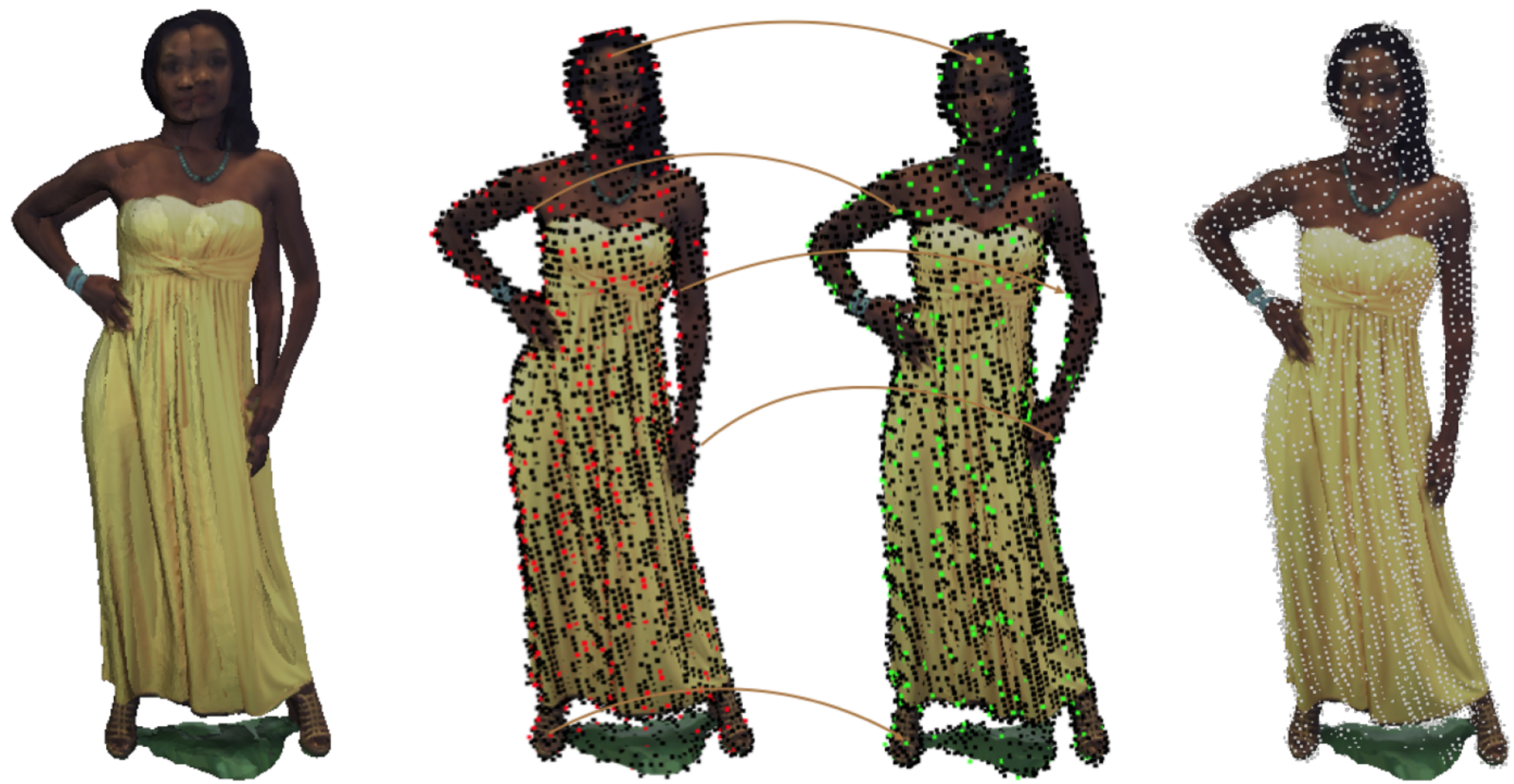
Graph SP representation used for motion estimation and compensation, and eventually predictive coding

Graph-based Motion Estimation



Graph SP representation used for motion estimation and compensation, and eventually predictive coding

Motion Compensation - Example



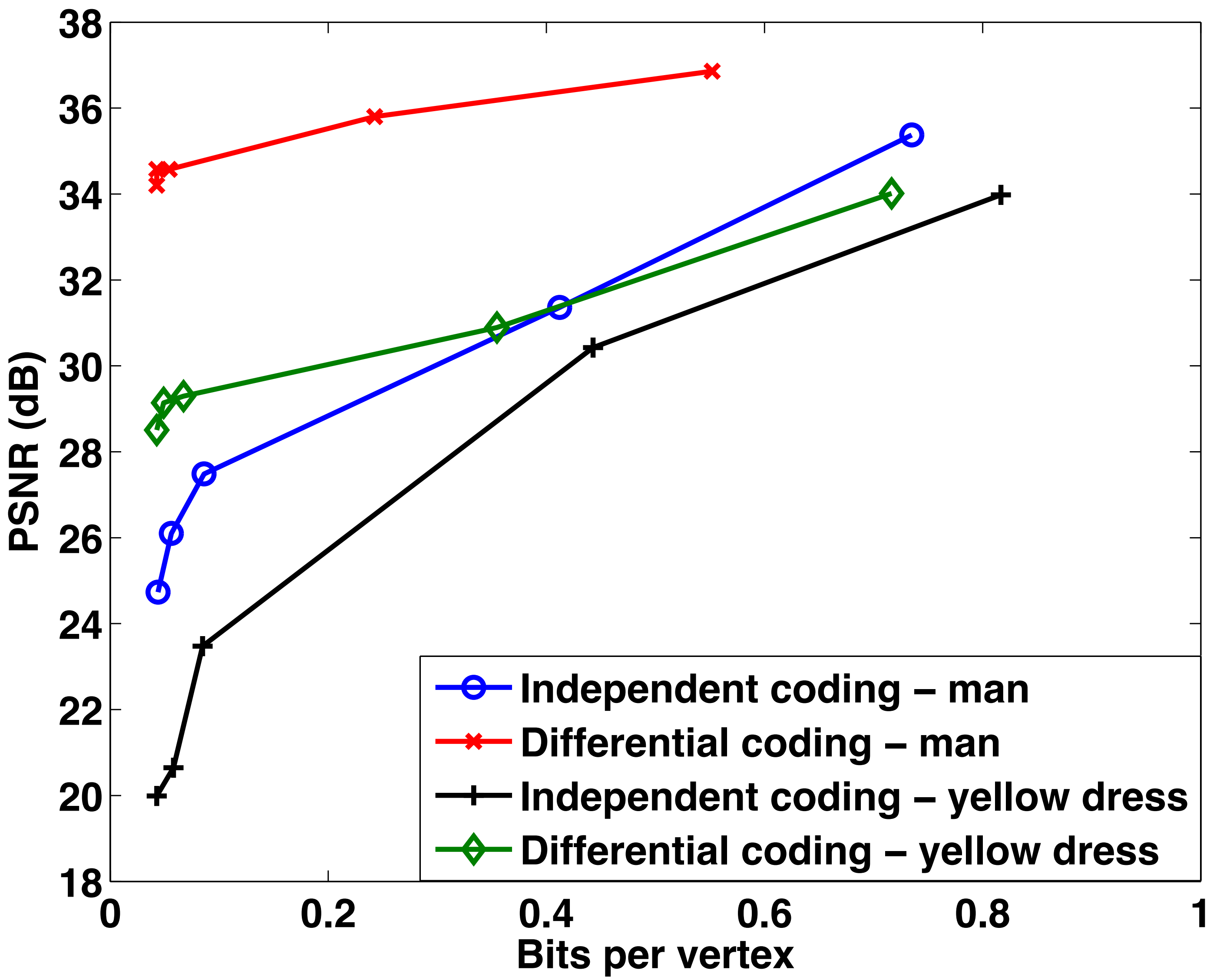
(a) reference + target frame

(b) sparse correspondences between frames

(c) motion compensated reference frame + target frame

- The sparse set of matching vertices are accurate and well-distributed in space

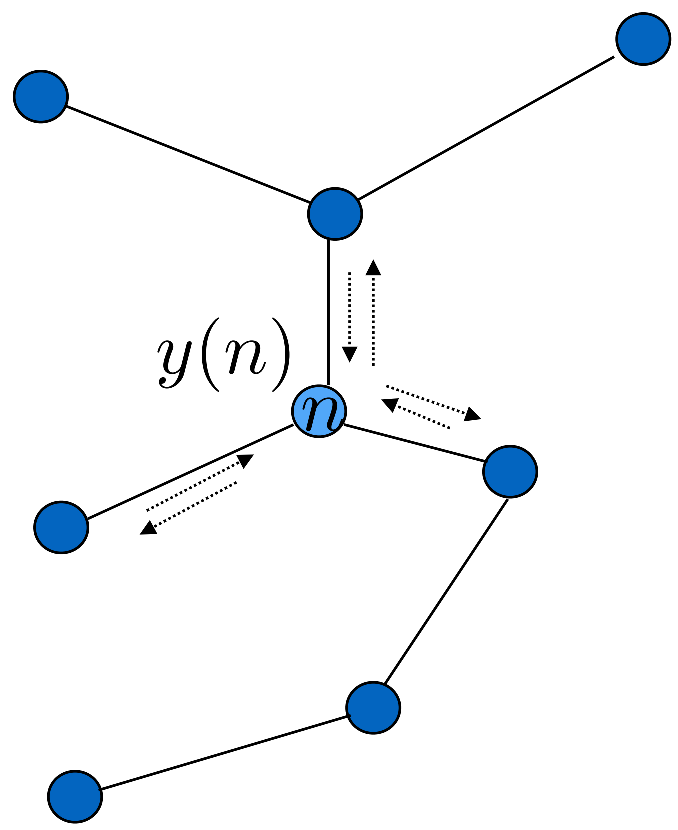
3D Color Compression Results



Compression of 10 frames in each sequence using predictive coding [Zhang:2014]

Distributed processing

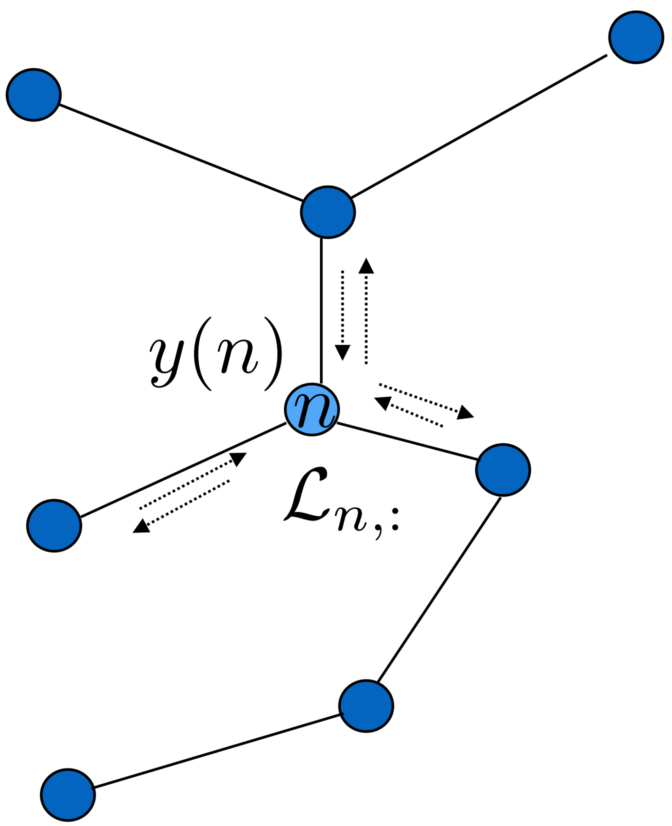
- Graph signal: function on a network
  - e.g., measurement in a wireless sensor network
- Signal processing tasks
  - denoising, reconstruction, inference
- Communication constraints
  - centralised processing is not possible
  - no node fully knows the signal
  - only local communication



Processing on graphs: denoising

- Denoising (LASSO) problem

$$x^* = \min_x \|y - \mathcal{D}x\|_2^2 + \kappa \; \|x\|_1$$



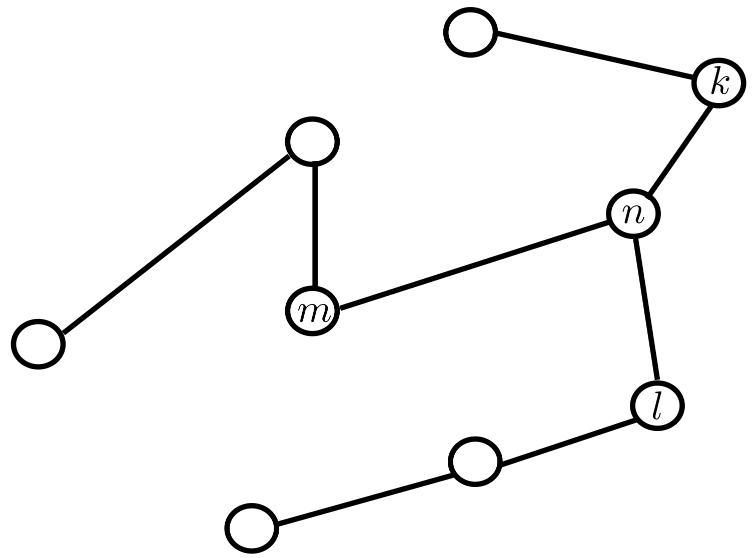
- Iterative soft thresholding solution

$$\mathcal{S}_{\kappa\tau} = \begin{cases} 0 & \text{if } |z| \leq \mu\tau \\ z - \operatorname{sgn}(z)\kappa\tau & \text{otherwise} \end{cases}$$

$$x^t = \mathcal{S}_{\kappa\tau}\left(x^{(t-1)} + 2\tau\mathcal{D}^T \; \left(y - \mathcal{D}x^{(t-1)}\right)\right), \; t = 1, 2, \ldots$$

- Distributed solution feasible as the dictionary-based operators can be distributed!

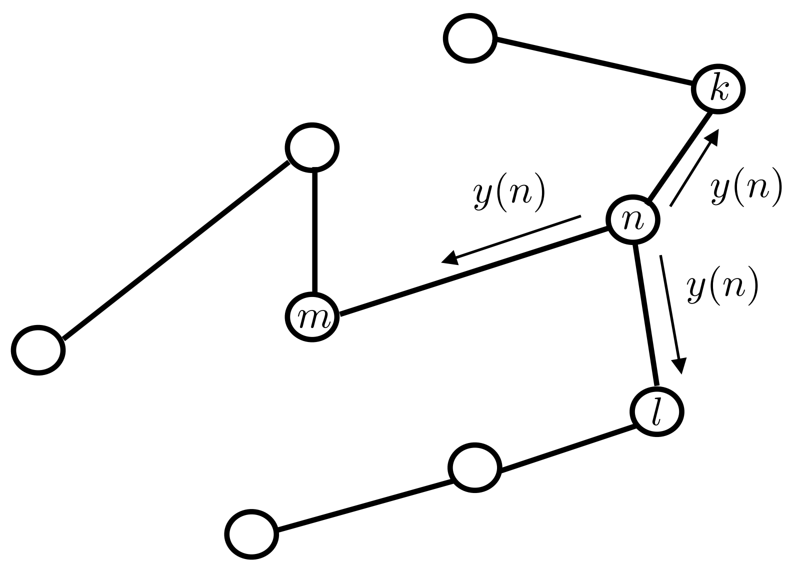
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

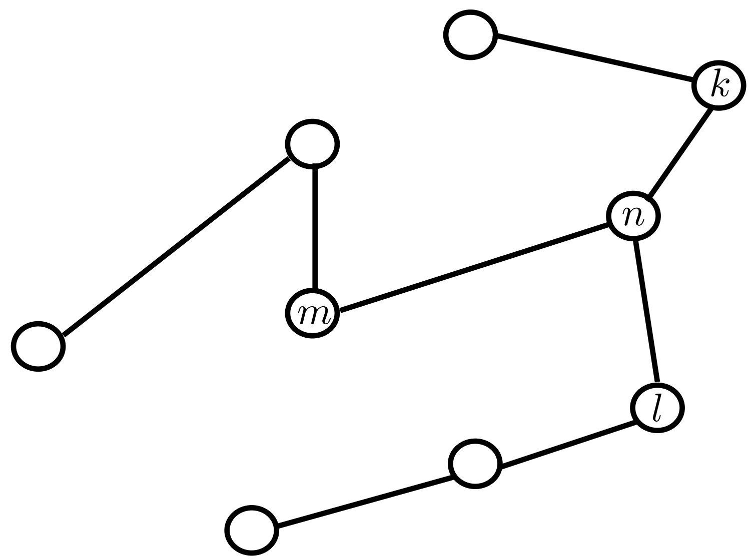
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

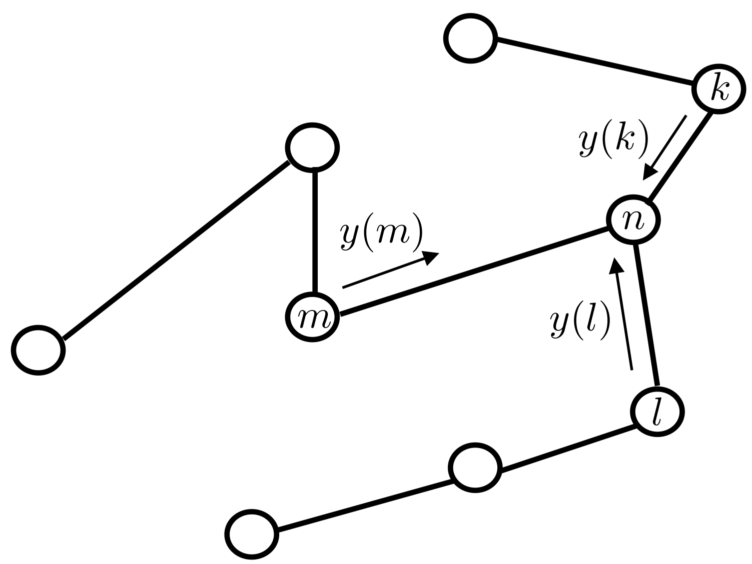
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

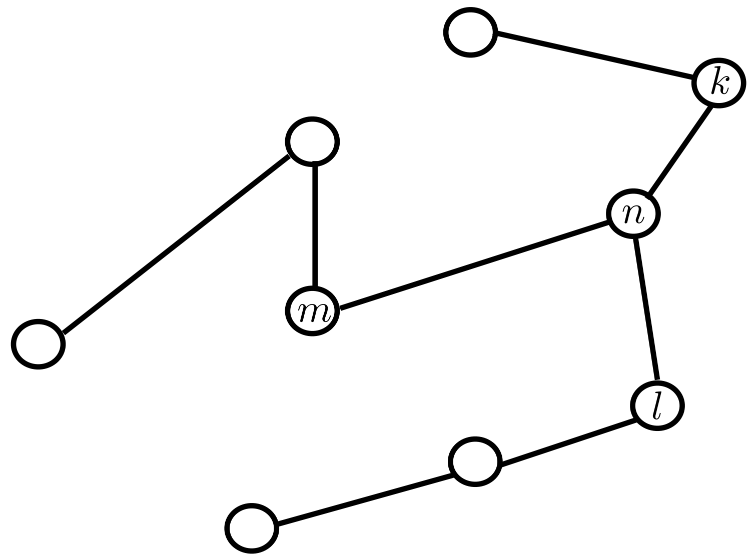
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

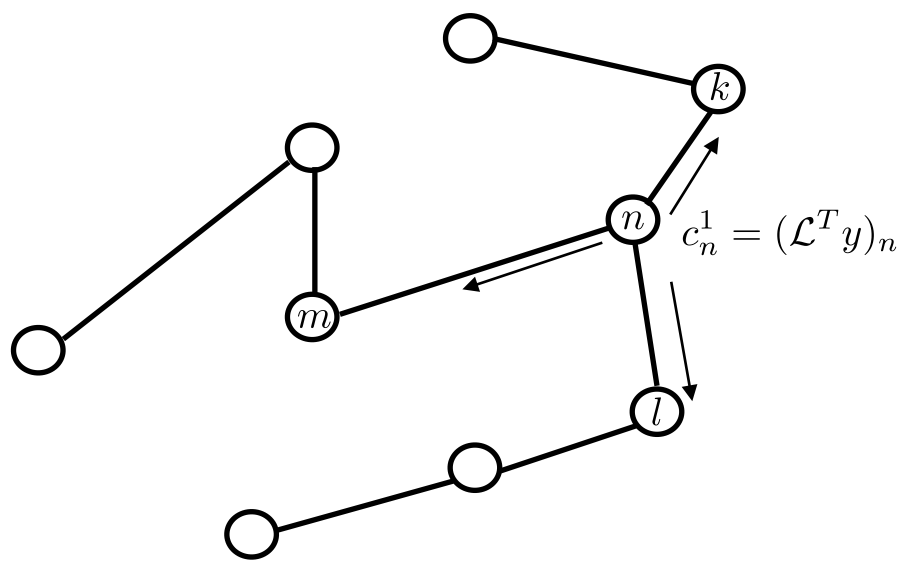
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

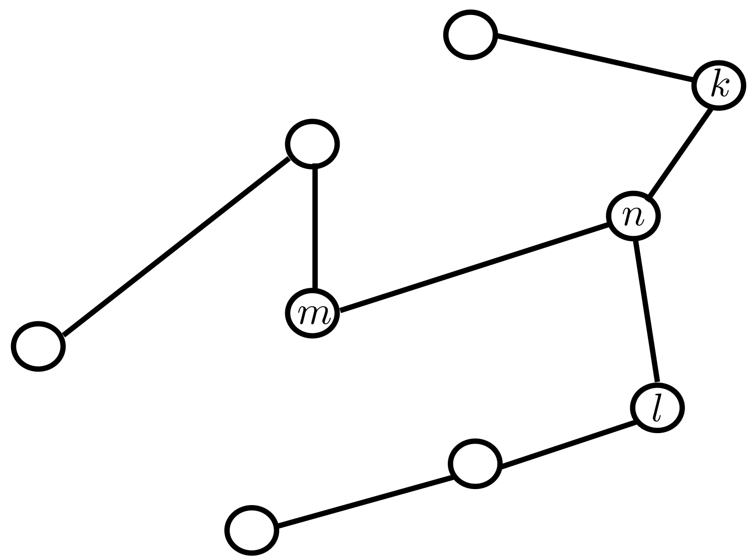
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^T y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^T y$

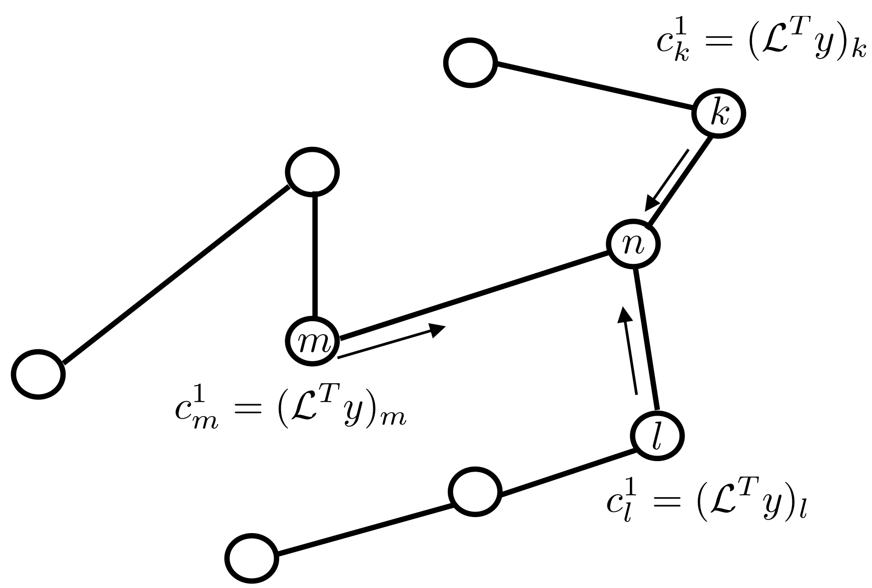
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

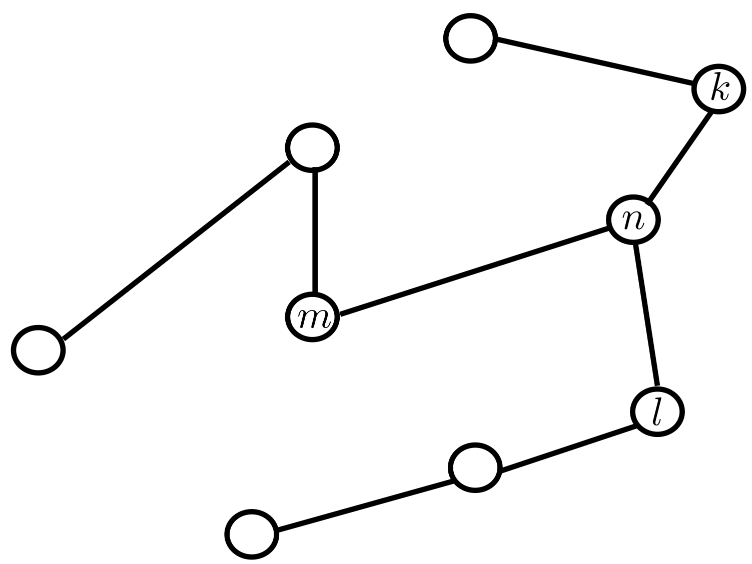
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

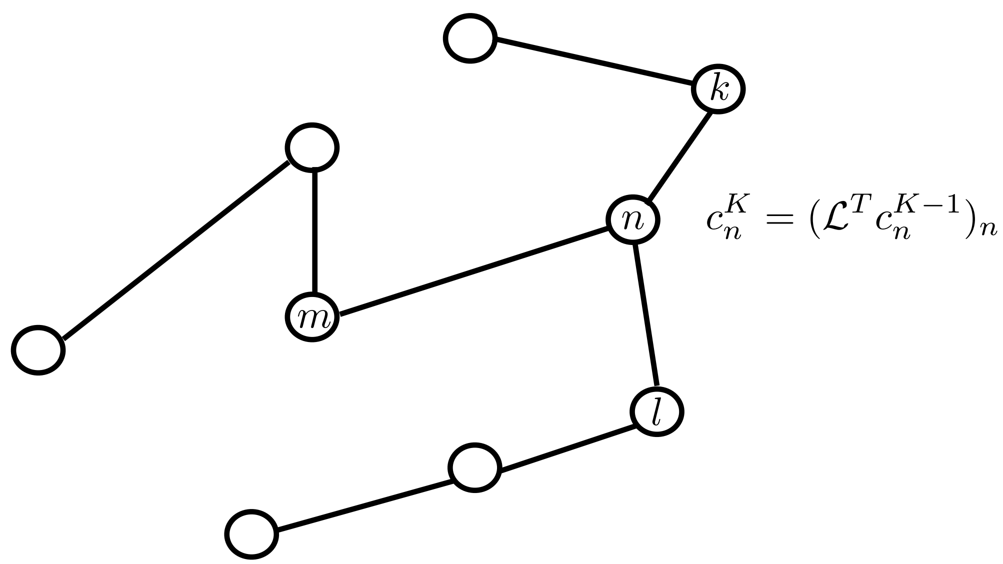
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^\mathcal{T} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^\mathcal{T} y$

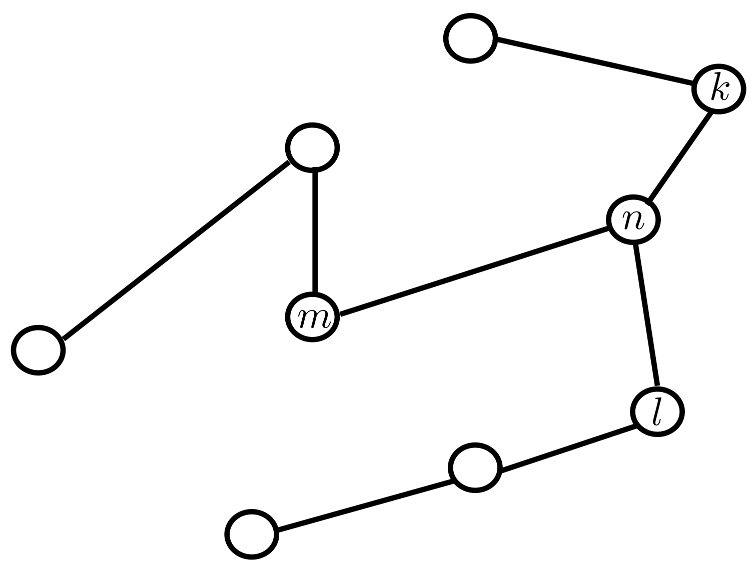
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^T y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^T y$

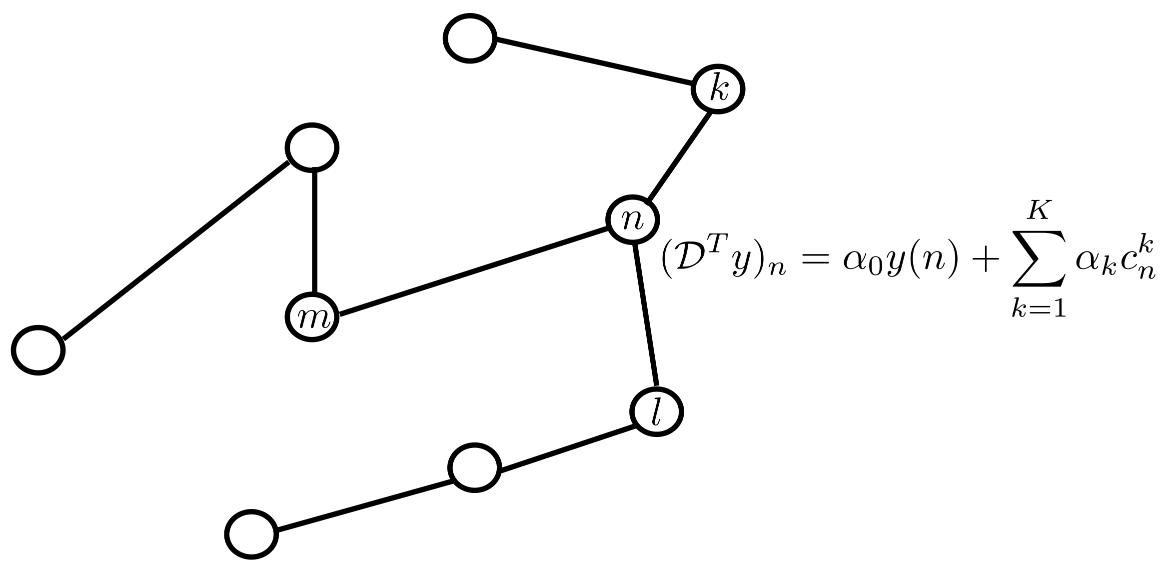
Illustration: adjoint operator



Distributed computation of $\mathcal{D}^{\mathcal{T}} y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^{\mathcal{T}} y$

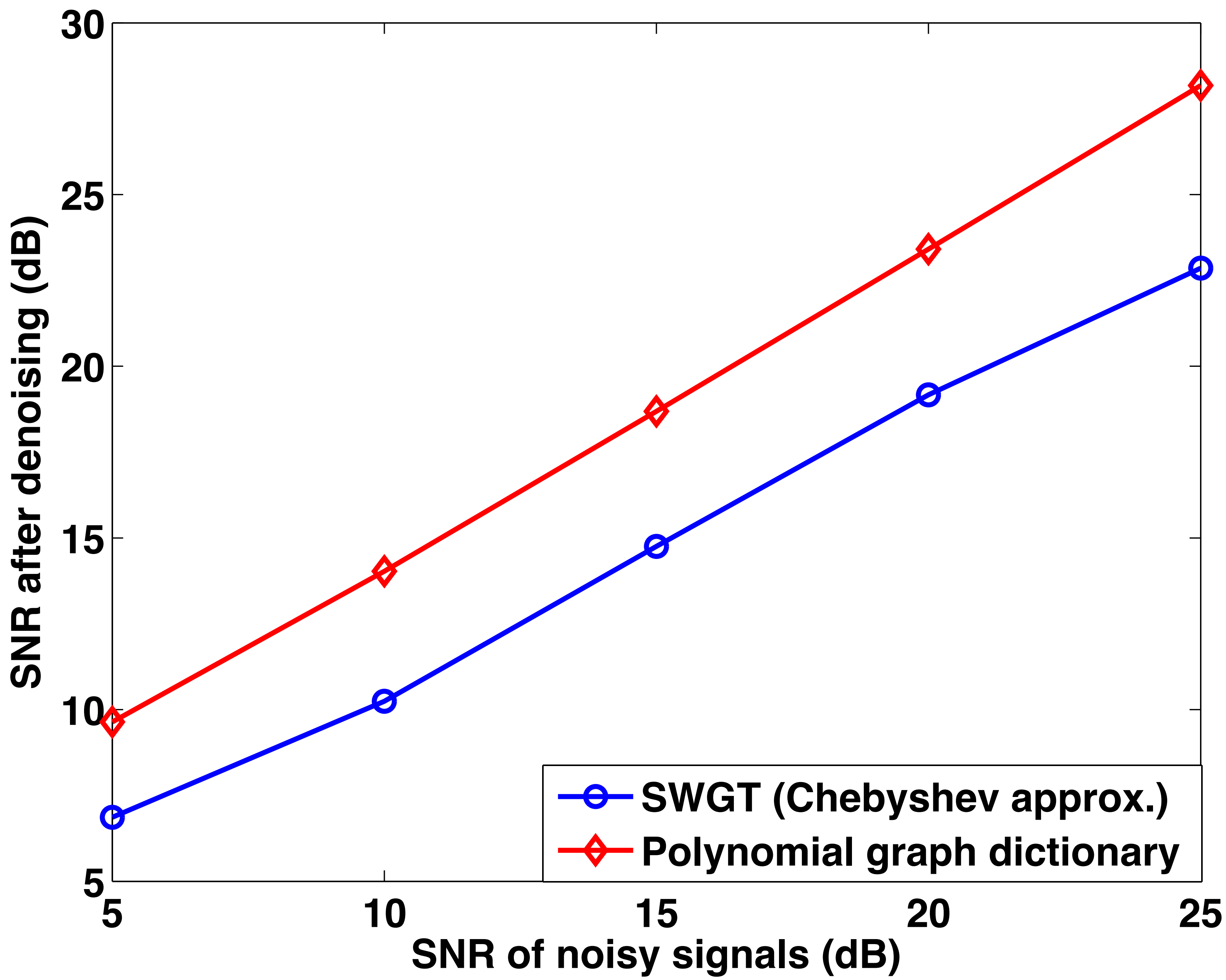
Illustration: adjoint operator



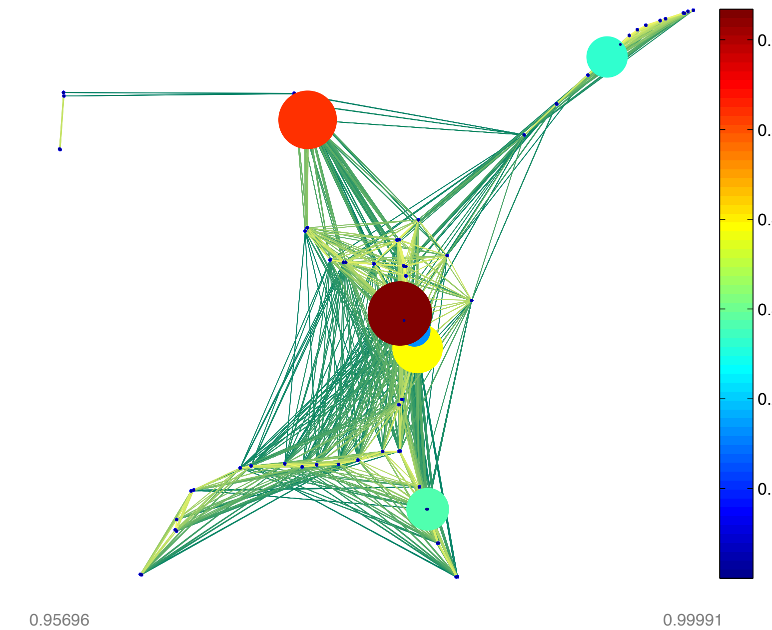
Distributed computation of $\mathcal{D}^T y$

Similar operations for the computation of $\mathcal{D}x$ and $\mathcal{D}\mathcal{D}^T y$

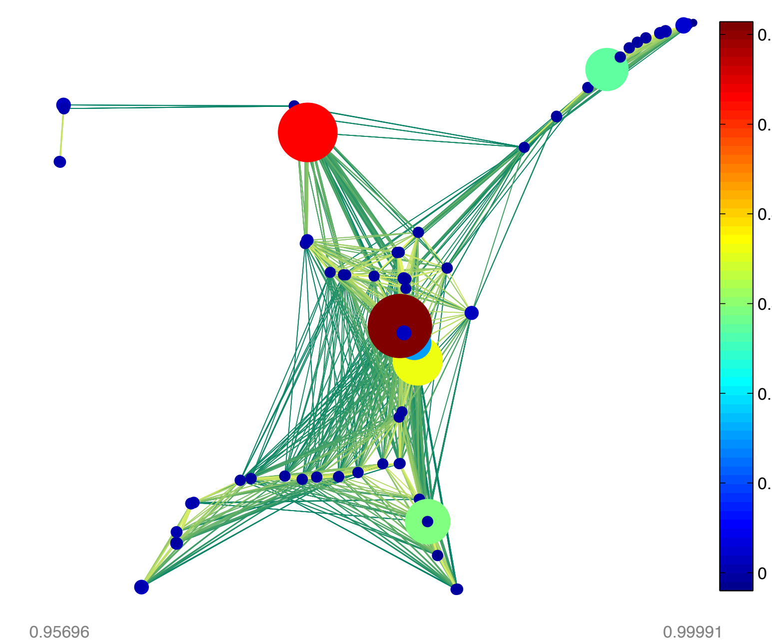
Denoising experiments



Distributed denoising with 100 ISTA iterations

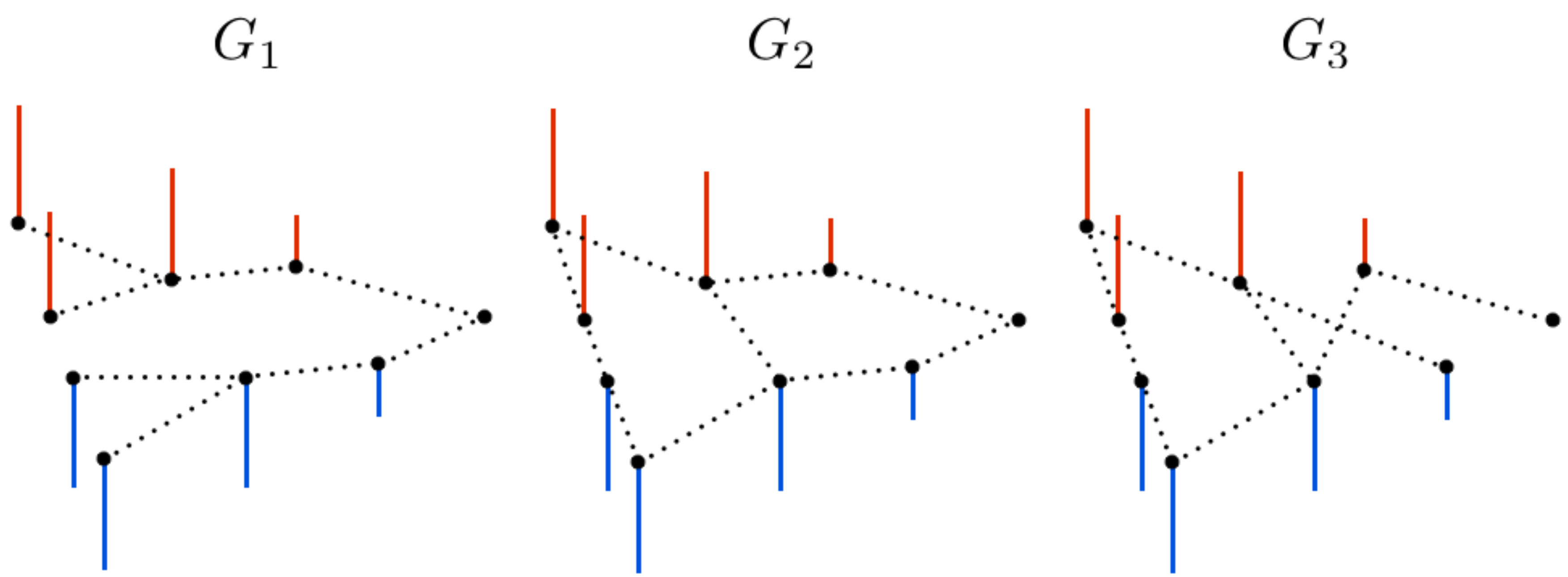


Clean traffic bottleneck signal

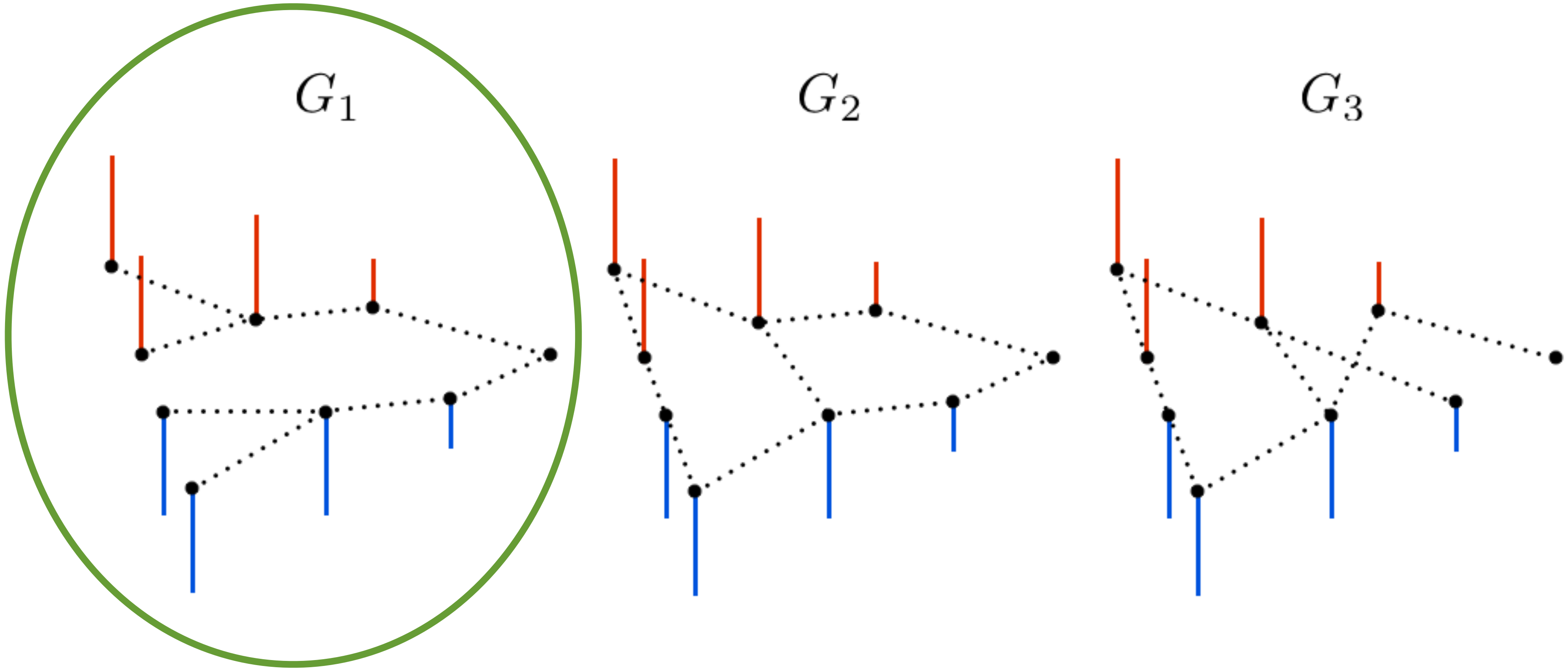


Denoised traffic bottleneck signal [24 dB]

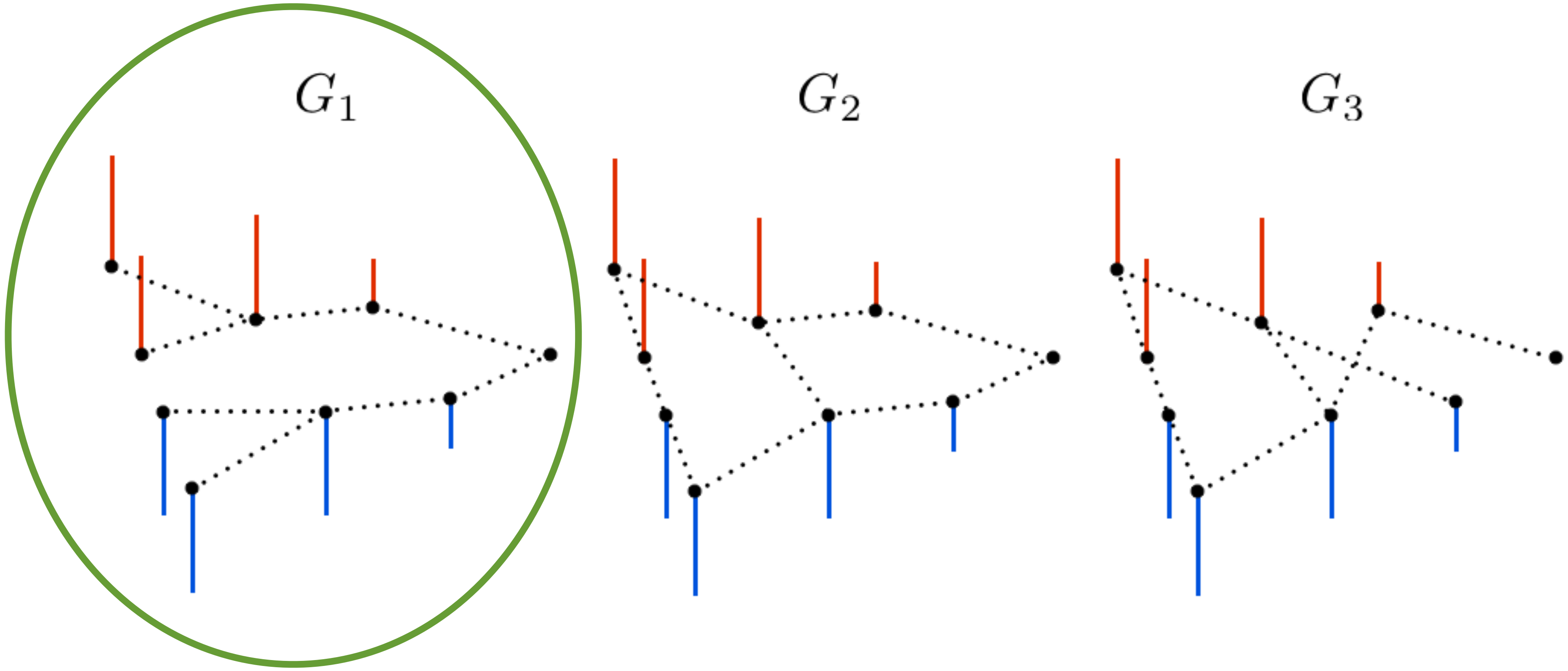
Joint signal and graph models



Joint signal and graph models



Joint signal and graph models

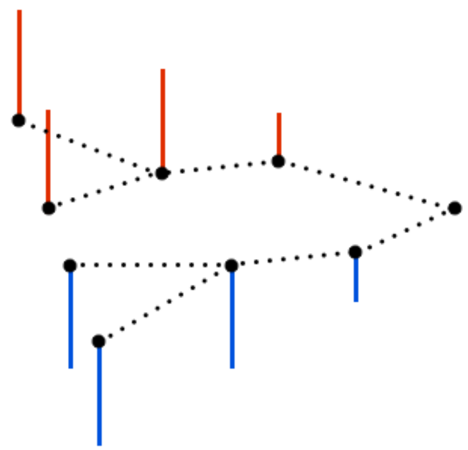


Challenge:

- How to define models of relationships between signals and graph?
- How to learn graphs that enforce desirable properties of graph signals?

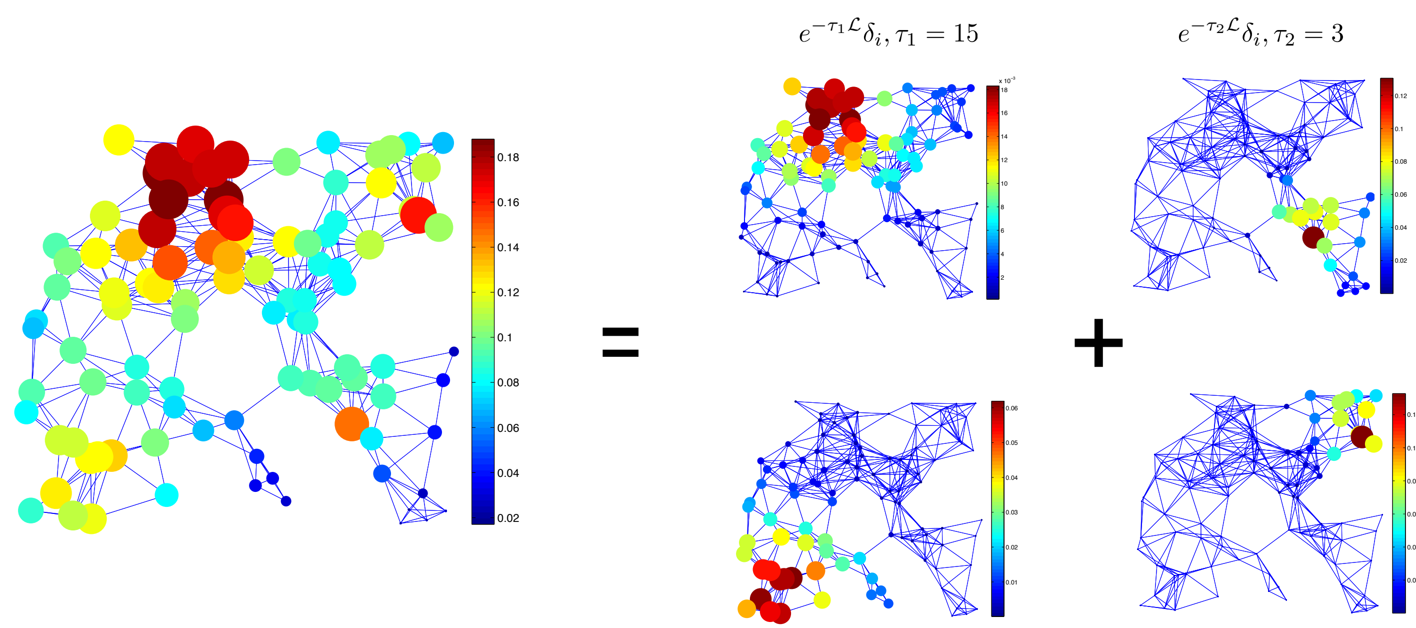
The importance of the graph

- Graph signal models define an interplay between signals and structure
  - Such models are used for effective data processing or analysis



smooth signal model

sparse signal model



- The graph might (often) not be known a priori
  - It becomes important to be able to infer the proper data structure

Graph learning: Beyond similarity

- Many ways to create the similarity graph…
  - Design it from the data (e.g., Gaussian RBF kernel)
  - Statistical approaches: covariance selection, probabilistic graphical models

Can we exploit the interplay between graphs and signals on graphs to discover the topology?

- Define graph signal representation models
- Use these models for recovering the graph: a very ill-posed problem

Graph learning: A GSP perspective

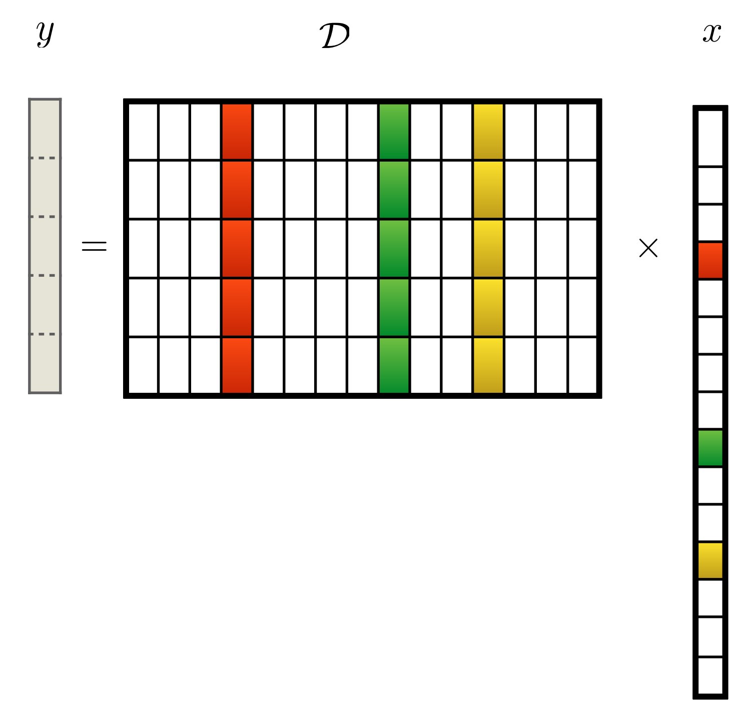
Gaussian noise: $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$

$$y = \mathcal{D}(L)x + \epsilon$$

observation

representation matrix: function of the graph

latent variable



- Generalization of the **factor analysis model on graphs**
  - Different priors on $\mathcal{D}$, x lead to different representations: learn the graph by imposing different priors (smooth and sparse)

Smoothness prior

- Set $\mathcal{D}(L) = \chi$, then

$$y = \chi x + \epsilon$$

eigenvectors of the graph Laplacian

Gaussian prior: $x \sim \mathcal{N}(0, \Lambda^{\dagger})$

- If $x \sim \mathcal{N}(0, \Lambda^{\dagger})$, then $y \sim \mathcal{N}(0, L^{\dagger} + \sigma_{\epsilon}^{2} I)$ (GMRF)
- The MAP estimator of

$$\begin{aligned} x_{\mathrm{MAP}}(y) &= \arg\max_{x} p(x|y) \\ &= \arg\min_{x} \left(-\log p_E(y - \chi x) - \log p_X(x)\right) \\ &= \arg\min_{x} \|y - \chi x\|_2^2 + \alpha x^T \Lambda x \end{aligned}$$

smoothness term

$$x^T \Lambda x = x^T \chi^T (\chi \Lambda \chi^T) \chi x = y^T L y$$

Graph learning for smooth representations

$$\min_{\chi,\Lambda,x} ||y - \chi x||_2^2 + \alpha \; x^T \Lambda x \quad \xrightarrow{z = \chi x} \quad \min_{L,z} ||y - z||_2^2 + \alpha \; z^T L z$$

Graph learning for smooth representations

$$\min_{L \in \mathbb{R}^{N \times N}, Z \in \mathbb{R}^{N \times p}} \quad ||Y - Z||_F^2 + \alpha \, \mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$

$$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

Graph learning for smooth representations

$$\min_{L \in \mathbb{R}^{N \times N}, Z \in \mathbb{R}^{N \times p}} ||Y - Z||_F^2 + \alpha \, \mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$

$$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \; i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

- Alternating minimization between:
  - Step 1: $$\min_{L \in \mathbb{R}^{N \times N}} \alpha \, \mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$
    $$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \; i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

Graph learning for smooth representations

$$\min_{L\in\mathbb{R}^{N\times N}, Z\in\mathbb{R}^{N\times p}} ||Y - Z||_F^2 + \alpha \ \mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$
$$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

- Alternating minimization between:
  - Step 1: $$\min_{L\in\mathbb{R}^{N\times N}} \alpha \ \mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$ $$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$
  - Step 2: $$\min_{Z\in\mathbb{R}^{N\times p}} ||Y - Z||_F^2 + \alpha \ \mathrm{tr}(Z^T L Z)$$

Graph learning for smooth representations

$$\min_{L\in\mathbb{R}^{N\times N}, Z\in\mathbb{R}^{N\times p}} ||Y-Z||_F^2 + \alpha \,\mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$

$$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\; i \neq j, \quad L\cdot\mathbf{1} = \mathbf{0}$$

- Alternating minimization between:
  - Step 1: $$\min_{L\in\mathbb{R}^{N\times N}} \alpha \,\mathrm{tr}(Z^T L Z) + \beta ||L||_F^2,$$ $$\text{s.t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\; i \neq j, \quad L\cdot\mathbf{1} = \mathbf{0}$$
  - Step 2: $$\min_{Z\in\mathbb{R}^{N\times p}} ||Y-Z||_F^2 + \alpha \,\mathrm{tr}(Z^T L Z)$$

Both steps are convex optimization problems

Graph learning for smooth representations

$$\min_{L\in\mathbb{R}^{N\times N}, Z\in\mathbb{R}^{N\times p}} ||Y - Z||_F^2 + \alpha \operatorname{tr}(Z^T L Z) + \beta ||L||_F^2,$$

$$\text{s.t.} \quad \operatorname{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

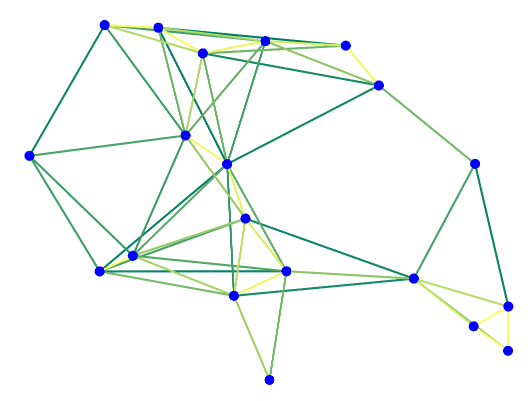
- Alternating minimization between:
  - Step 1: $$\min_{L\in\mathbb{R}^{N\times N}} \alpha \operatorname{tr}(Z^T L Z) + \beta ||L||_F^2,$$ $$\text{s.t.} \quad \operatorname{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$
  - Step 2: $$\min_{Z\in\mathbb{R}^{N\times p}} ||Y - Z||_F^2 + \alpha \operatorname{tr}(Z^T L Z)$$

Both steps are convex optimization problems

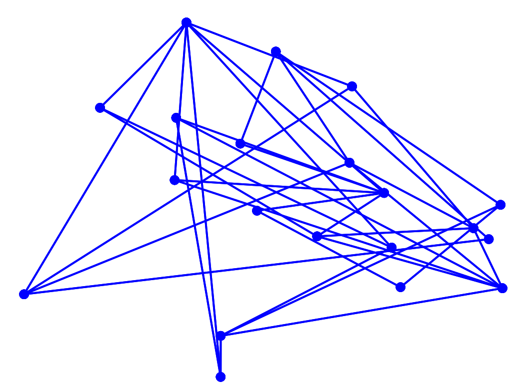
[Dong:TSP2015] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning Laplacian matrix in smooth graph signal representations,” Submitted to IEEE Trans. Signal Process., 2015

Synthetic signals

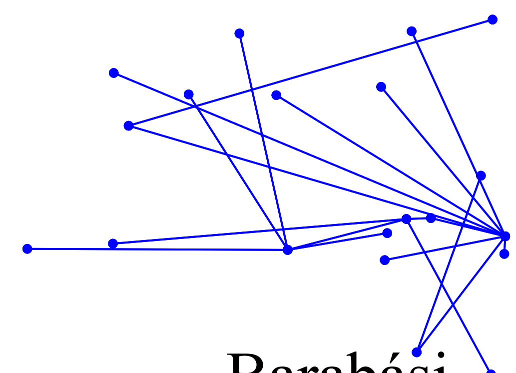
- Generate random graphs based on three models
- Generate graph signals that follow Gaussian distributions with mean zero and precision matrix being graph Laplacians
- Learn graphs using only the signals



Gaussian RBF

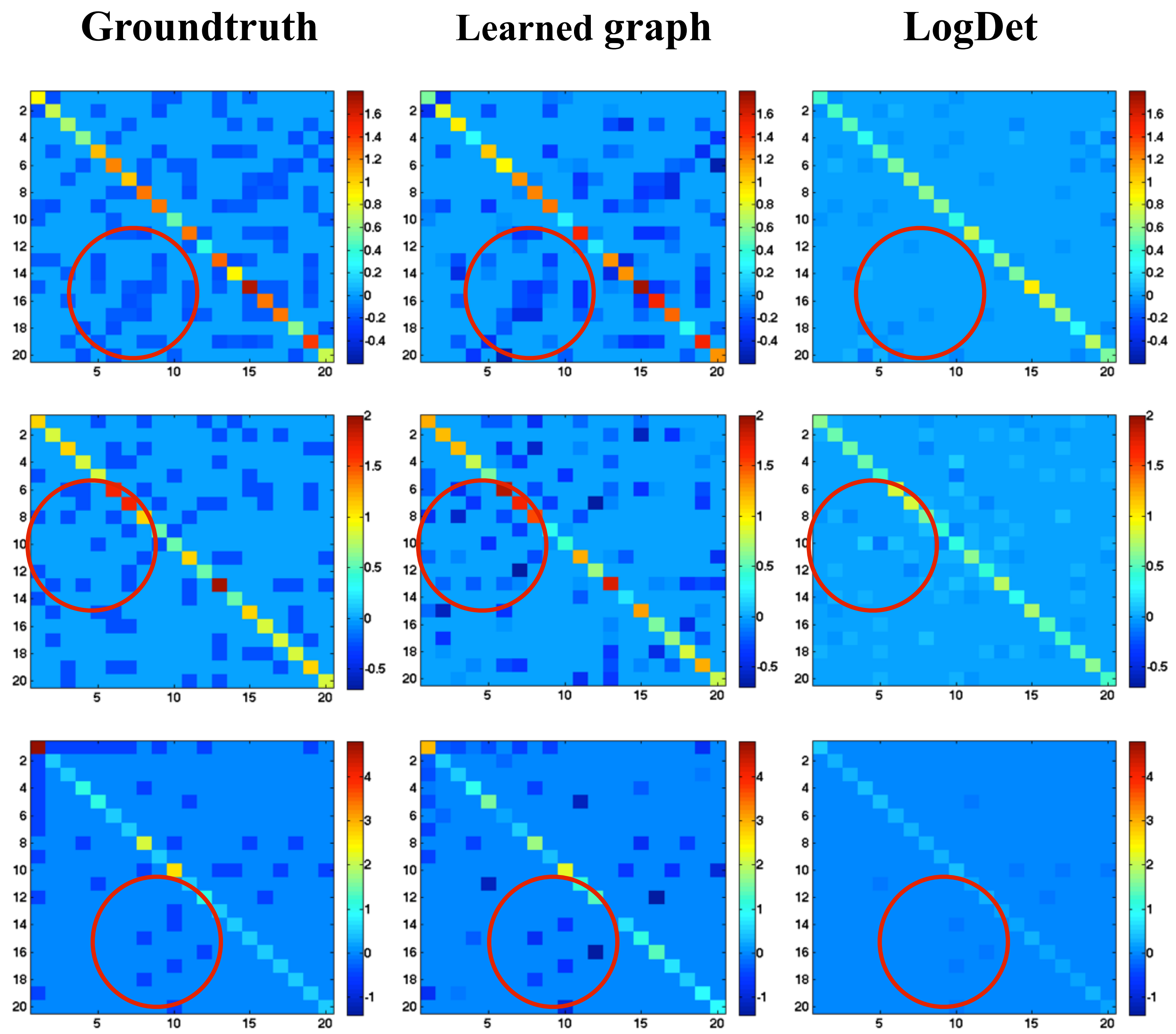


Erdős-Rényi



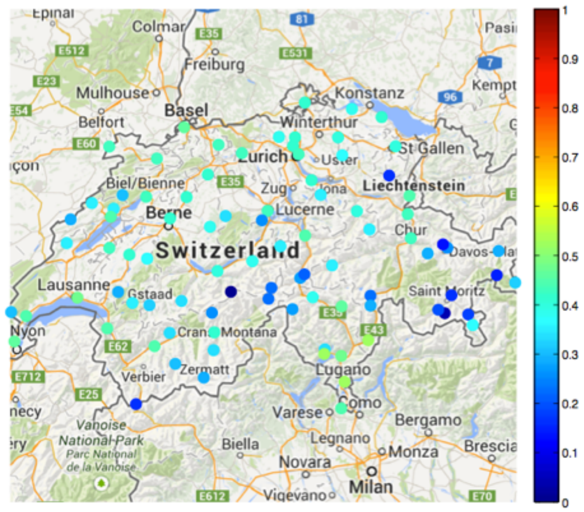
Barabási-Albert

Results: Synthetic signals

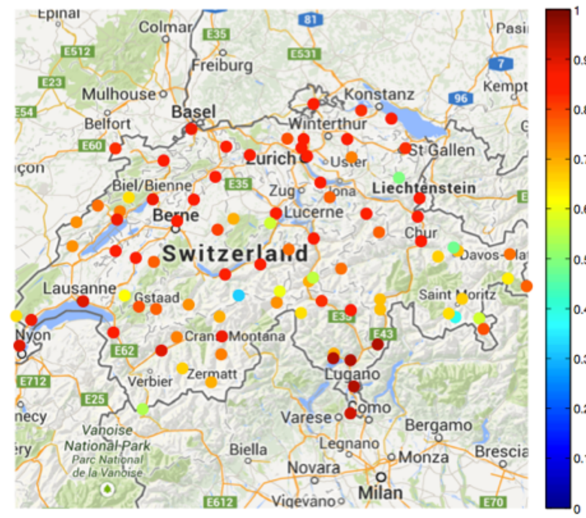


Learning meteorological graph

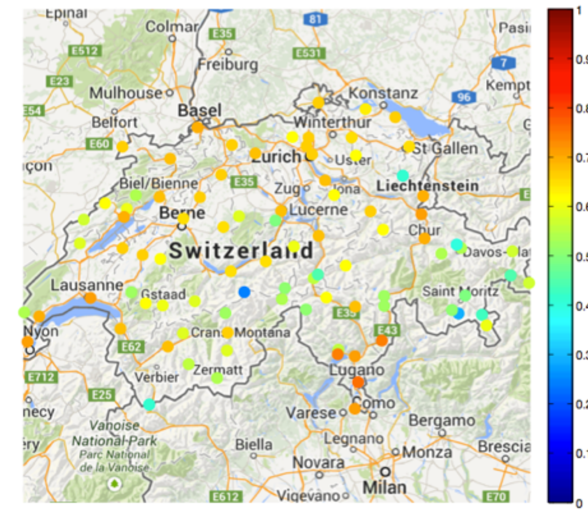
February



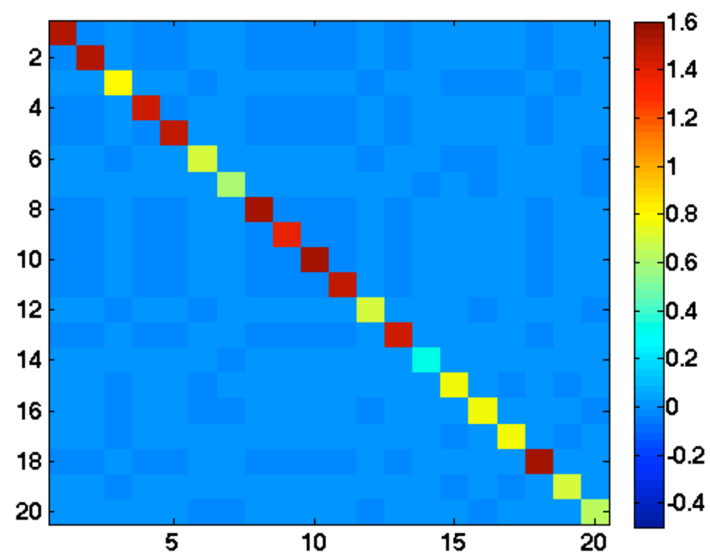
June



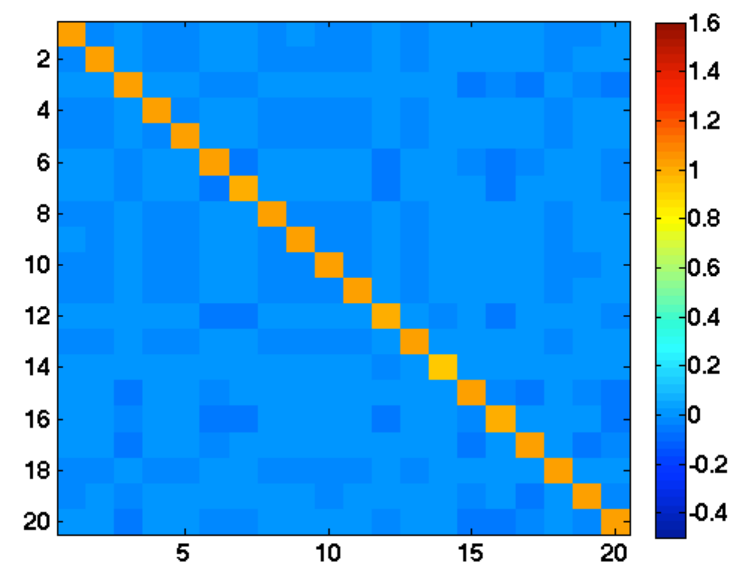
October



Groundtruth

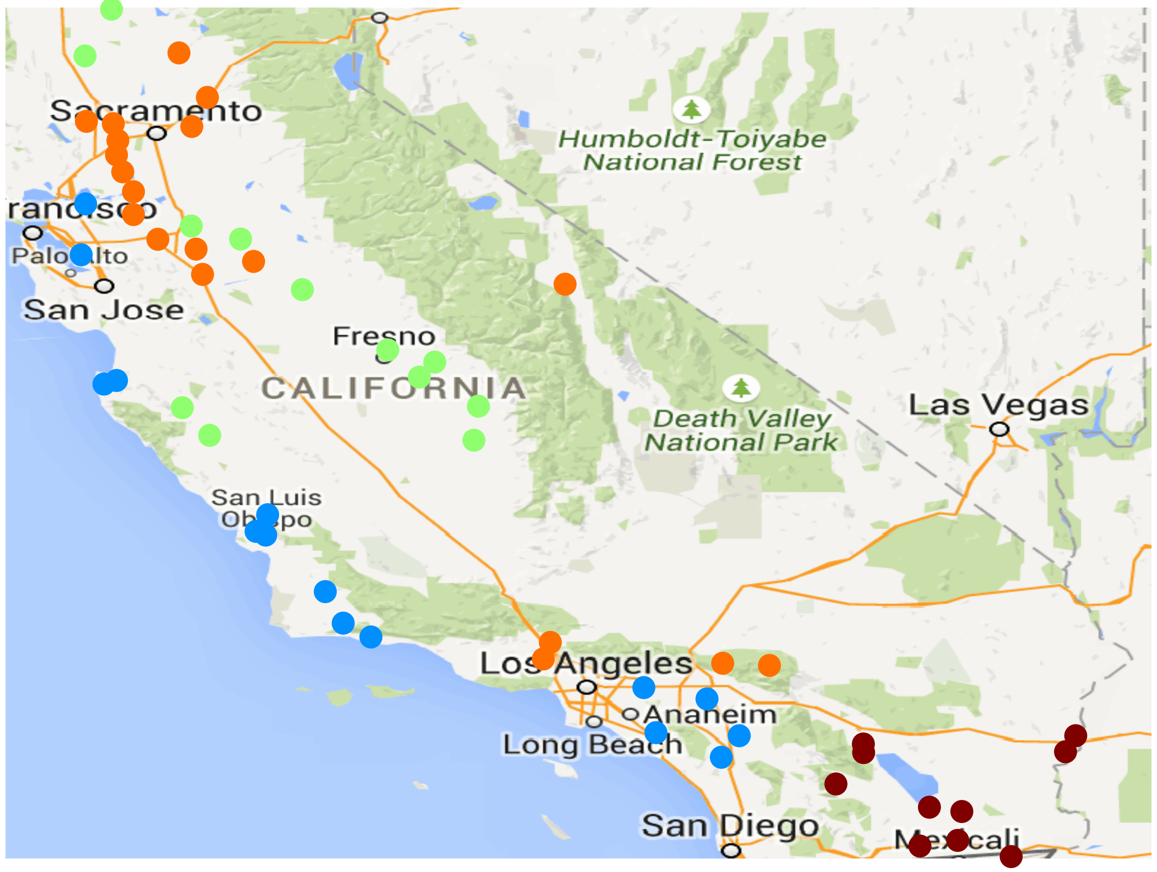


Learned graph



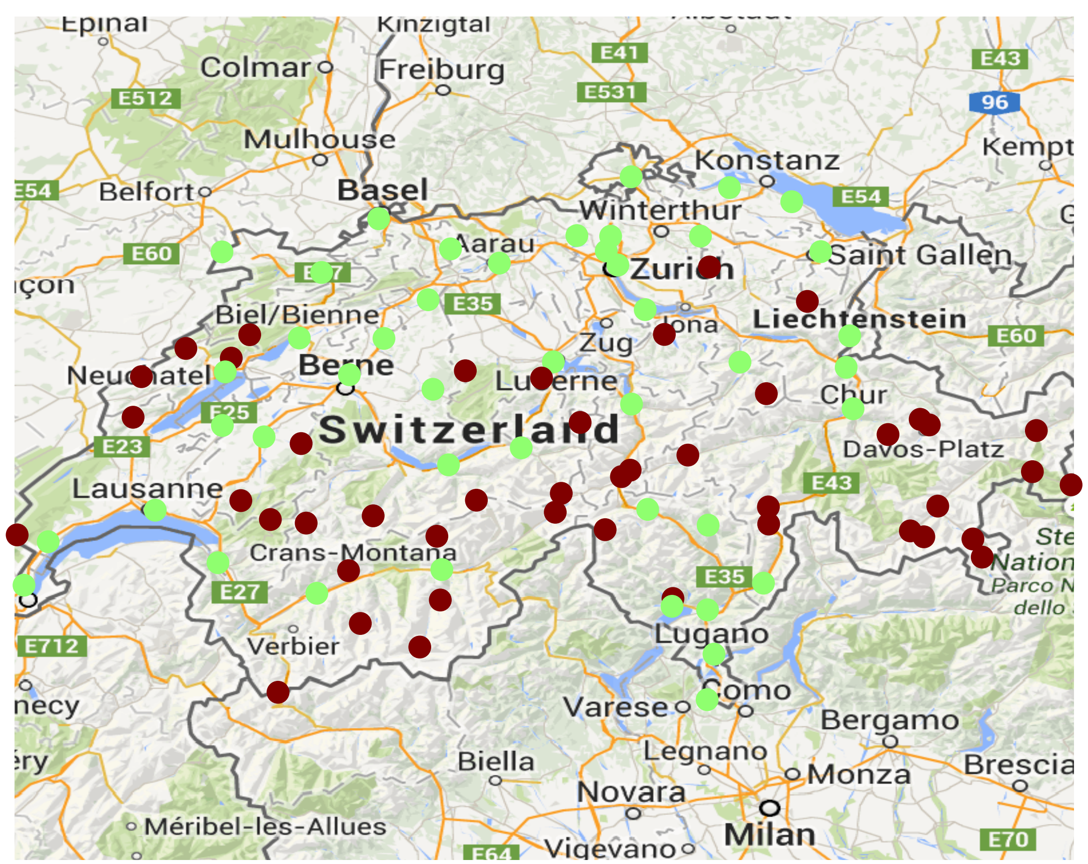
Analysis of meteorological data

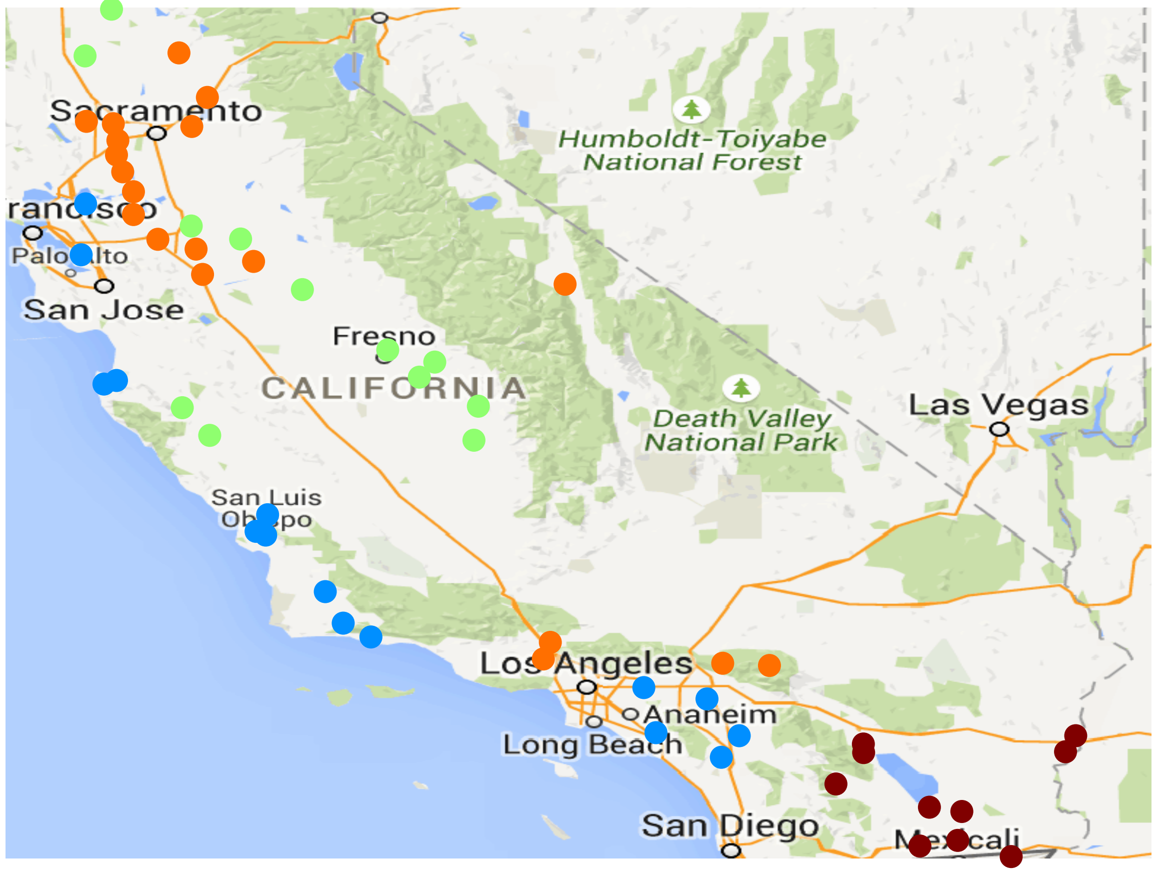
- The learned graph can be used for partitioning entities into several clusters, e.g., [Ng, NIPS'01]



Analysis of meteorological data

- The learned graph can be used for partitioning entities into several clusters, e.g., [Ng, NIPS'01]



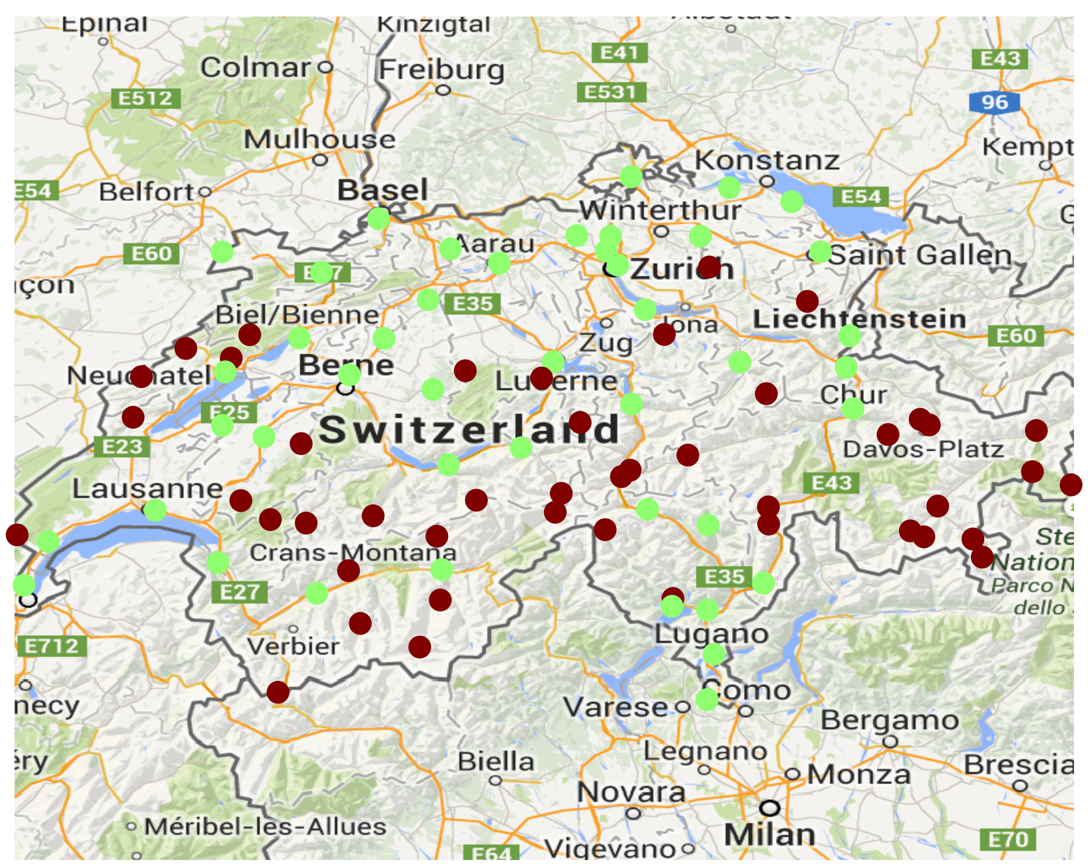


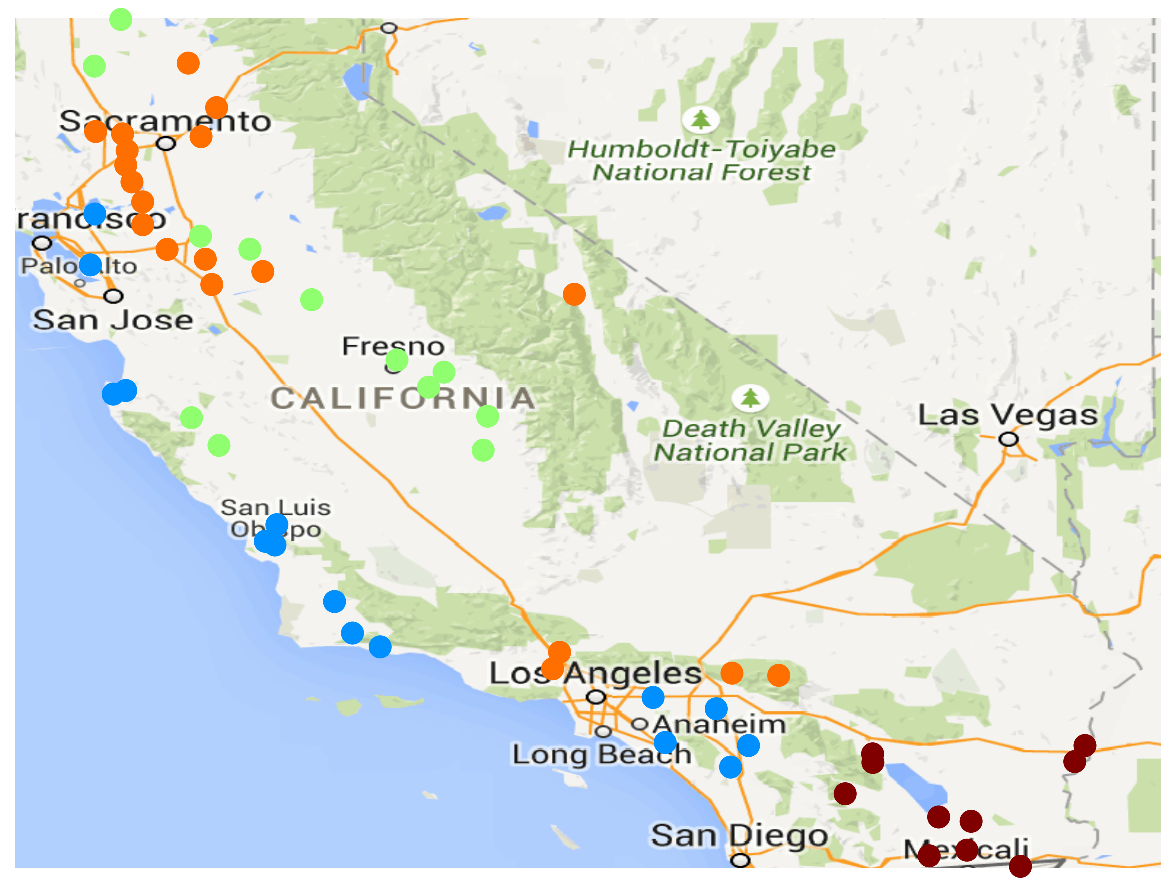
Mountain regions (Jura, Alps)

Flat regions and valleys

Analysis of meteorological data

- The learned graph can be used for partitioning entities into several clusters, e.g., [Ng, NIPS'01]

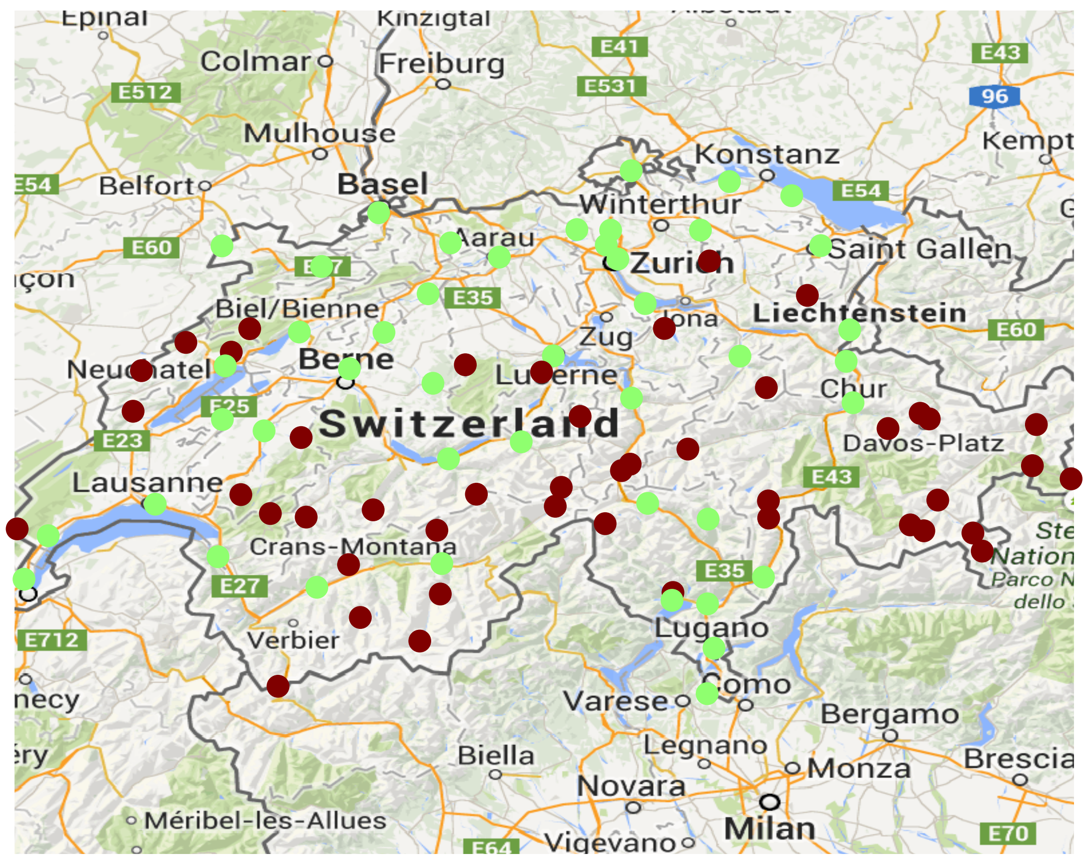


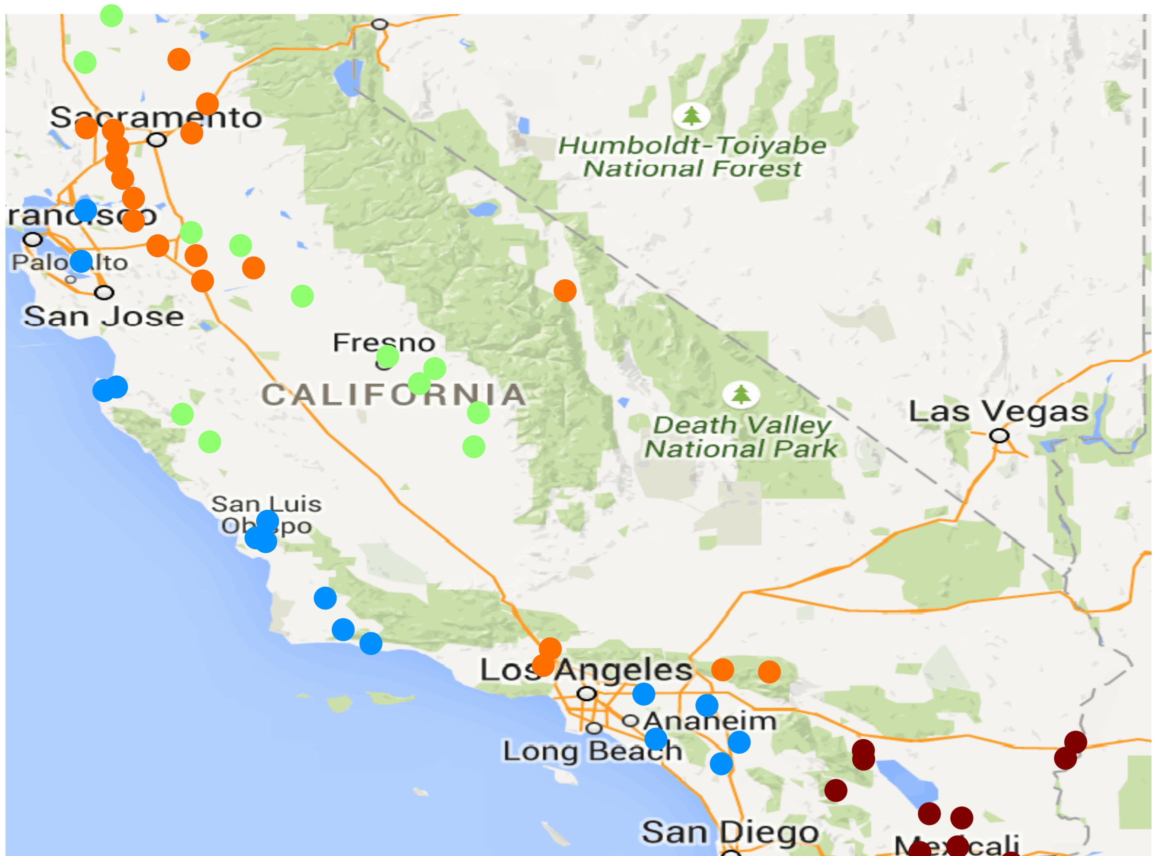


4 ETo zones

Analysis of meteorological data

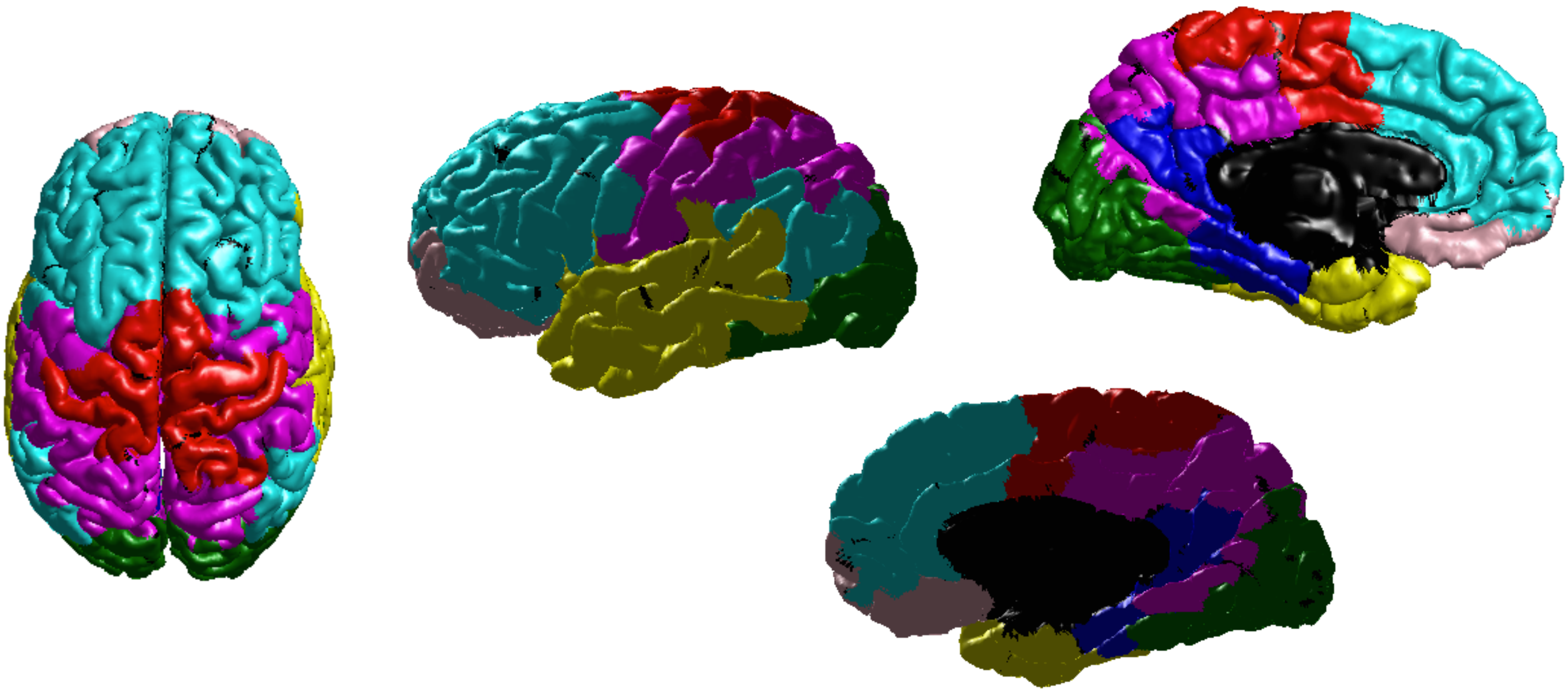
- The learned graph can be used for partitioning entities into several clusters, e.g., [Ng, NIPS'01]





Infer brain connections

- Signals: time series recorded in MRI scan while the subjects are at rest



Parcellation of the brain obtained with the learned graph

Sparse prior

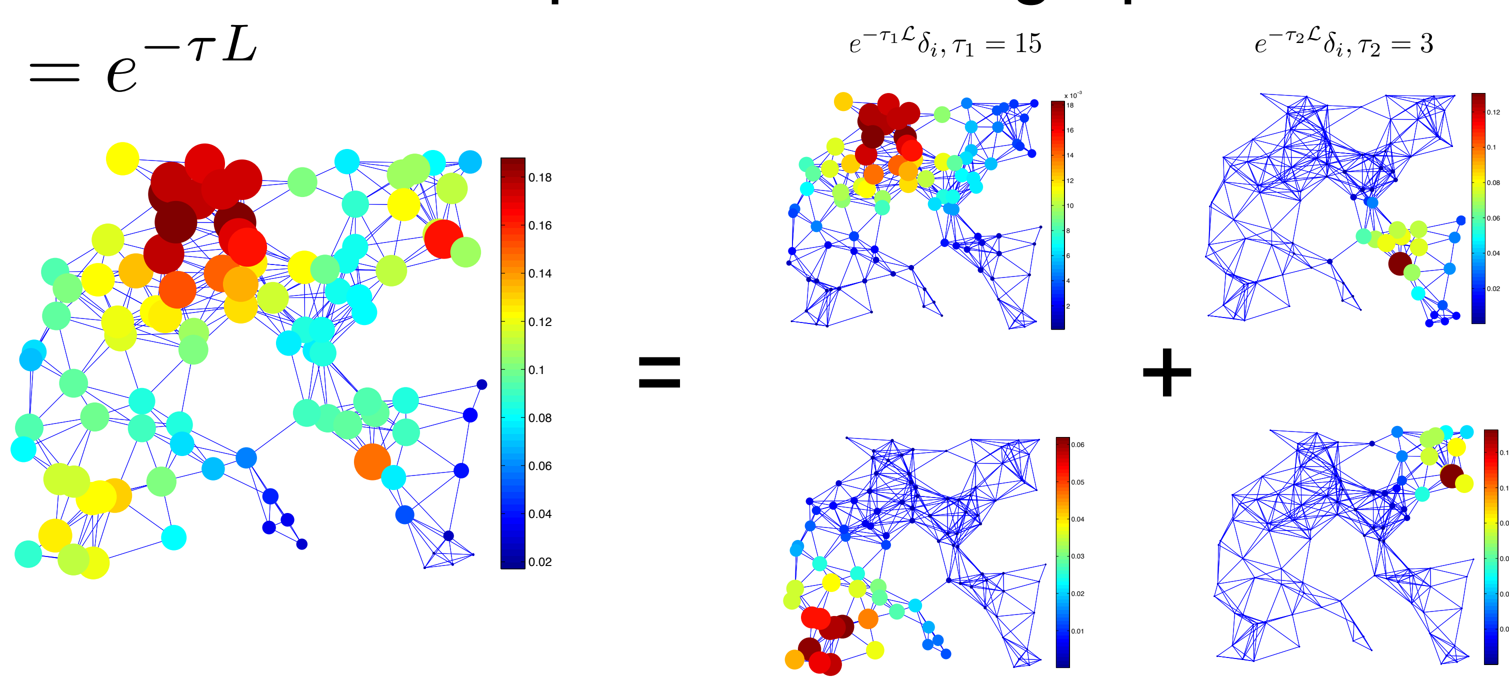
$$y = g(L)x + \epsilon$$

filters of the graph Laplacian

Laplace prior:
$$p(x) = \prod \alpha \exp(-\alpha |x_i|)$$

- Model for diffusion phenomena/localized processes on graph

    - **heat kernel** $g(L) = e^{-\tau L}$



Sparse graph signal representations

$$y = g(L)x + \epsilon$$

filters of the graph Laplacian

Laplace prior:
$p(x) = \prod \alpha \exp(-\alpha |x_i|)$

- Model for diffusion phenomena/localized processes on graph
  - **heat kernel** $g(L) = e^{-\tau L}$

$$x_{\mathrm{MAP}}(y) = \arg\max_x p(x|y) = \arg\max_x p(y|x)p(x)$$

$$= \arg\min_x \left(-\log\, p_E(y - e^{-\tau L}x) - \log\, p_x(x)\right)$$

$$= \arg\min_x ||y - e^{-\tau L}x||_2^2 + \alpha ||x||_1$$

Graph learning for sparse signals

$$\min_{x,L} ||y - e^{-\tau L}x||_2^2 + \alpha||x||_1$$

Graph learning for sparse signals

$$\min_{X,L} ||Y - e^{-\tau L} X||_2^2 + \alpha \sum_{j=1}^{p} ||x_j||_1$$

Graph learning for sparse signals

$$\min_{L,\ X} \ ||Y - e^{-\tau L} X||_F^2 + \alpha \sum_{j=1}^{p} ||x_j||_1 + \beta \|L\|_F^2$$

$$\text{s. t.} \quad \text{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

Graph learning for sparse signals

$$\min_{L,\ X} \ ||Y - e^{-\tau L} X||_F^2 + \alpha \sum_{j=1}^{p} ||x_j||_1 + \beta \|L\|_F^2$$

$$\text{s. t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

- Solve using PALM [Bolte:MathProg2014]

$$Z(L,X) = \|Y - e^{-\tau L} X\|_F^2, \quad f(X) = \sum_{j=1}^{p} \|x_j\|_1, \quad g(L) = \delta(L|\mathcal{C}) + \beta \|L\|_F^2,$$

Graph learning for sparse signals

$$\min_{L,\ X} \ ||Y - e^{-\tau L} X||_F^2 + \alpha \sum_{j=1}^{p} ||x_j||_1 + \beta \|L\|_F^2$$

$$\text{s. t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0,\ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

- Solve using PALM [Bolte:MathProg2014]

$$Z(L, X) = \|Y - e^{-\tau L} X\|_F^2, \quad f(X) = \sum_{j=1}^{p} \|x_j\|_1, \quad g(L) = \delta(L|\mathcal{C}) + \beta \|L\|_F^2,$$

convex set

$$\mathcal{C} = \{\mathrm{tr}(L) = N, L_{ij} = L_{ji} \leq 0,\ i \neq j, L \cdot \mathbf{1} = \mathbf{0}\}$$

convex indicator function

$$\delta(L|\mathcal{C}) = \begin{cases} 1, & \text{if } L \in \mathcal{C} \\ +\infty, & \text{otherwise.} \end{cases}$$

Graph learning for sparse signals

$$\min_{L,\ X} \ ||Y - e^{-\tau L} X||_F^2 + \alpha \sum_{j=1}^{p} ||x_j||_1 + \beta \|L\|_F^2$$

$$\text{s. t.} \quad \mathrm{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \ i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

- Solve using PALM [Bolte:MathProg2014]

$$Z(L, X) = \|Y - e^{-\tau L} X\|_F^2, \quad f(X) = \sum_{j=1}^{p} \|x_j\|_1, \quad g(L) = \delta(L|\mathcal{C}) + \beta \|L\|_F^2,$$

convex set

$$\mathcal{C} = \{\mathrm{tr}(L) = N, L_{ij} = L_{ji} \leq 0, \ i \neq j, L \cdot \mathbf{1} = \mathbf{0}\}$$

convex indicator function

$$\delta(L|\mathcal{C}) = \begin{cases} 1, & \text{if } L \in \mathcal{C} \\ +\infty, & \text{otherwise.} \end{cases}$$

[Thanou:NIPS2016] D. Thanou, X. Dong, D. Kressner and P. Frossard, “LearnHeat: A framework for learning heat diffusion graphs,” Submitted to NIPS, May 2016

PALM update steps

- Alternating minimization between:
  - Update of X:

$$X^{t+1} = \underset{X}{\operatorname{argmin}} \; < X - X^t, \nabla Z_H(L^t, X^t) > + \frac{c_t}{2}\|X - X^t\|^2 + f(X)$$

$$\Rightarrow \quad X^{t+1} = prox^{f}_{c_t}(X^t - \frac{1}{c_t}\nabla_X Z(L^t, X^t))$$

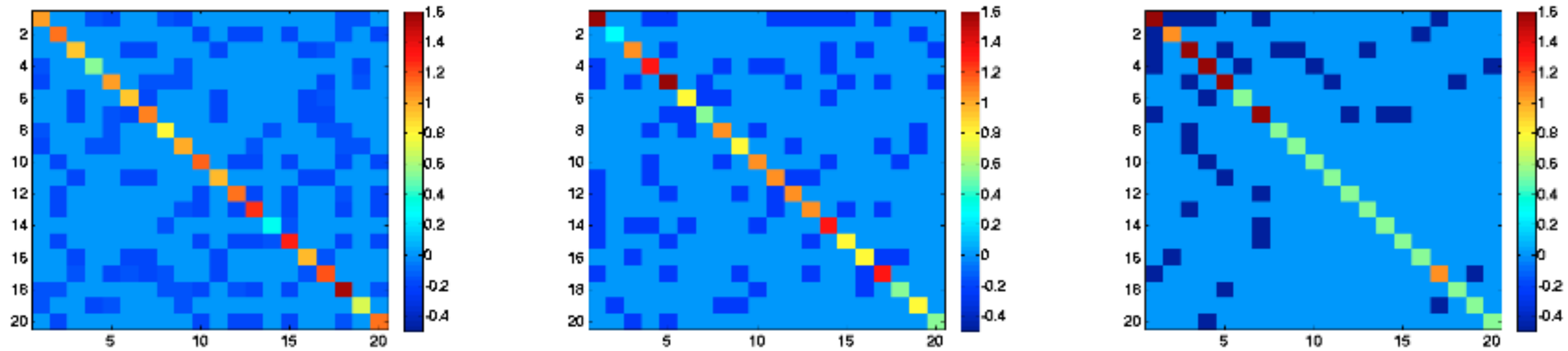
  - Update of L:

$$L^{t+1} = \underset{L}{\operatorname{argmin}} \quad < L - L^t, \nabla_L Z(X^{t+1}, L^t) > + \frac{d_k}{2}||L - L^t||^2_F + \beta\|L\|^2_F$$

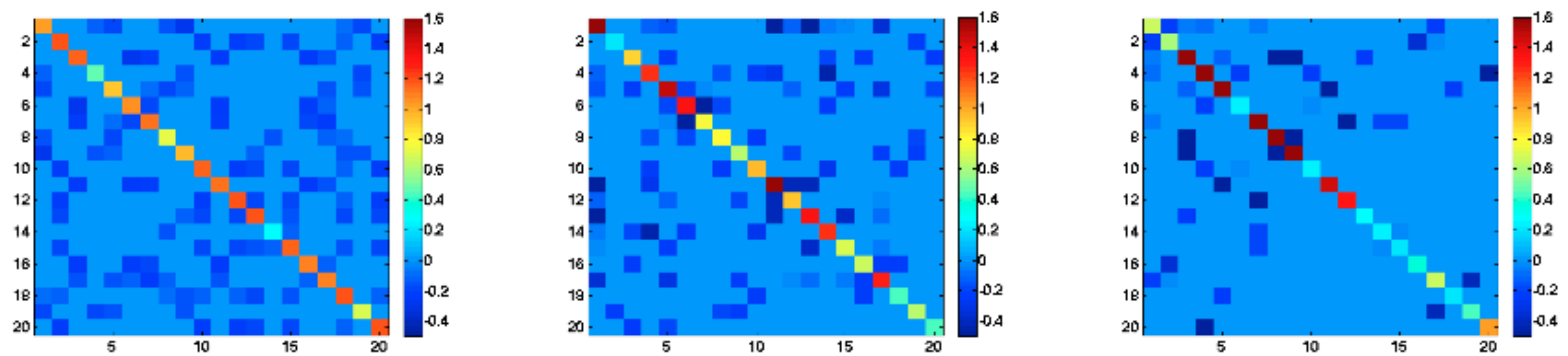
$$\text{s.t.} \quad \operatorname{tr}(L) = N, \quad L_{ij} = L_{ji} \leq 0, \; i \neq j, \quad L \cdot \mathbf{1} = \mathbf{0}$$

$$\Rightarrow \quad L^{t+1} = prox^{g}_{d_t}(L^t - \frac{1}{d_t}\nabla_L Z(L^t, X^{t+1}))$$

Experiments: Recovery performance



(a) Gaussian RBF: Groundtruth (b) ER: Groundtruth (c) BA: Groundtruth

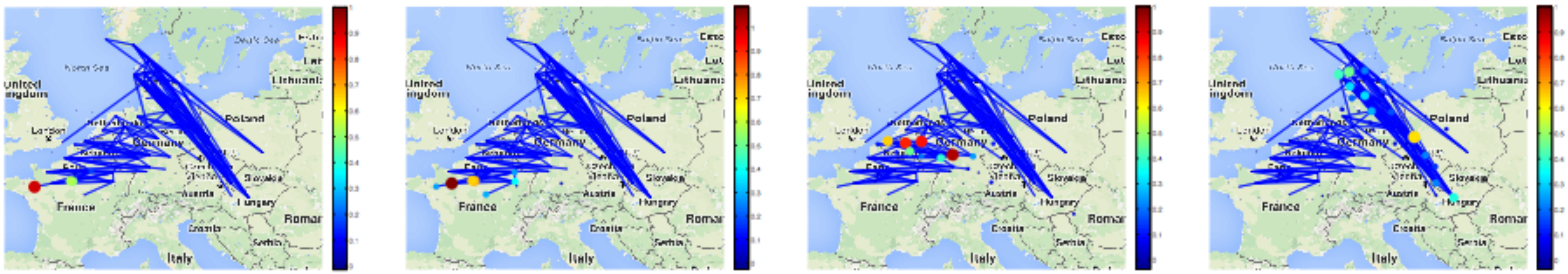


(d) Gaussian RBF: **GL-Laplace** (e) ER: **GL-Laplace** (f) BA: **GL-Laplace**

- Accurate recovery of the underlying graph from synthetic graph signals

Diffusion signals: ETEX dataset

- A tracer is released on the atmosphere from Rennes
- The concentration of the tracer over time is measured at 168 stations

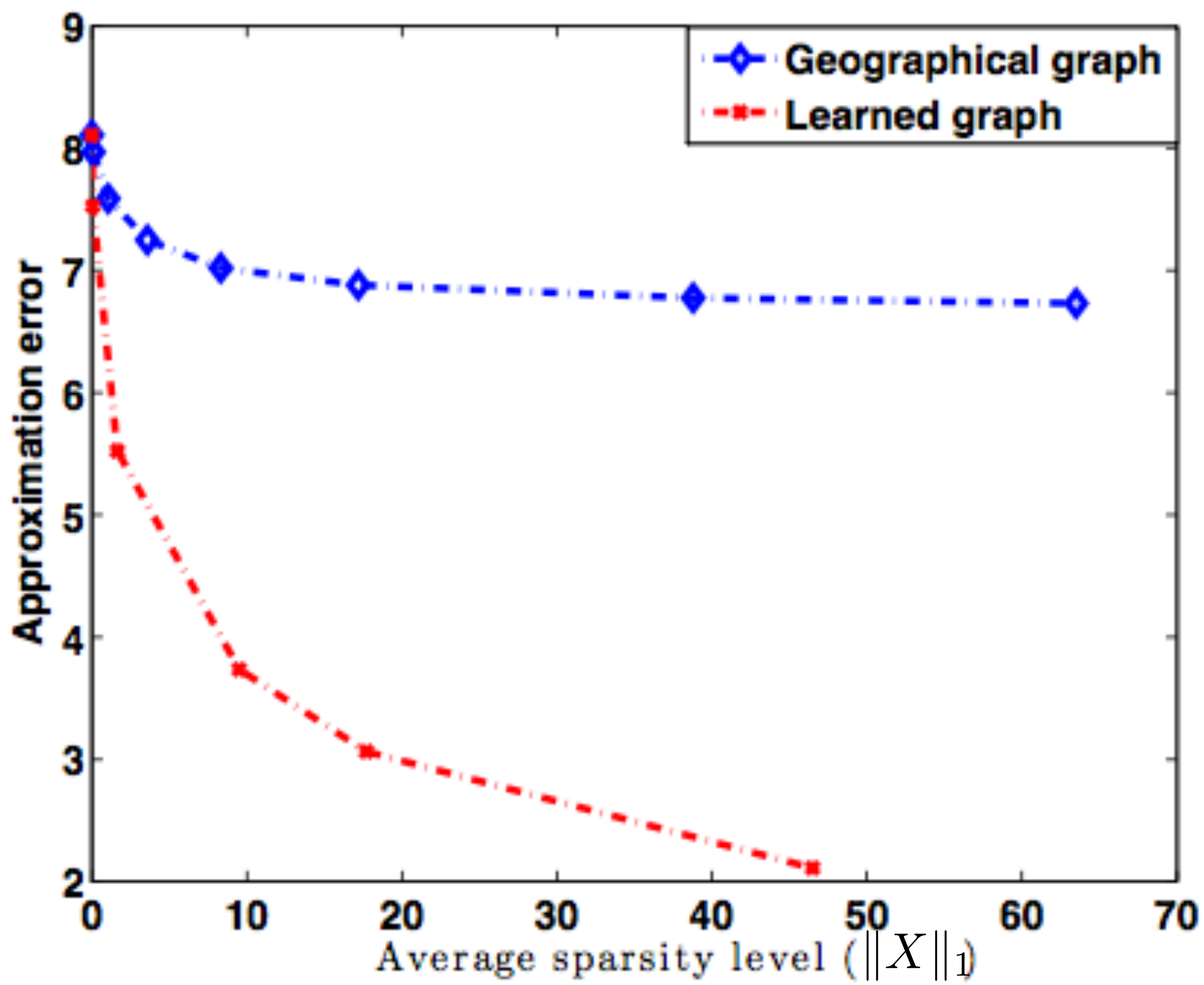


(a) (b) (c) (d)

- The learned graph indicates the main directions towards which the tracer moved that are consistent also with the graph signals

Application: Sparse representation

- The learned graph is used to generate a diffusion dictionary $g(L) = e^{-\tau L}$



  - It provides much sparser representation for the signals than a diffusion dictionary generated by the geographical graph

Summary

- Graph Signal Processing: joint consideration of the signal and the structure
- Structured adaptive representations lead to computationally effective operators
- In many problems, the graph is not known!

Quite a few open challenges :)

References

- D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high - dimensional data analysis to networks and other irregular domains," IEEE Signal Process. Mag., vol. 30, no. 3, pp. 83–98, May 2013
- D. Thanou, D. I Shuman, and P. Frossard, "Learning Parametric Dictionaries for Signals on Graphs'', IEEE Trans. Signal Process., vol. 62, no. 15, Aug. 2014
- D. Thanou, and P. Frossard, "Multi-graph learning of spectral graph dictionaries", IEEE ICASSP 2015 (best student paper award).
- X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in Proc. IEEE Int. Conf. Acc., Speech, and Signal Process., Kyoto, Japan, Mar. 2012, pp. 3373 – 3376.
- D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," IEEE Transactions on Image Processing, April 2016
- X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations," submitted to IEEE Transactions on Signal Processing, 2015
- D. Thanou and P. Frossard, "Distributed Signal Processing with Graph Spectral Dictionaries", Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing, UIUC, IL, USA, October 2015.